Deakin University
Faculty of Science, Engineering and Built Environment
School of IT



On a Multi–objective Optimiser: Choquet Adaptive Scalarisation - CAS

SIT316 – Linear and Nonlinear Optimisation

Lecturer: A/Prof. Julien Ugon

Supervisor: Prof. Gleb Beliakov

Student Name: Bao Minh Tran

Student ID: s224236373

Email: s224236373@deakin.edu.au

September 12^{th}, 2026

Abstract

We present a novel nonlinear multi–objective optimiser, which belongs to the family of adaptive scalarisation methods, and refer to it as Choquet Adaptive Scalarisation (CAS). We utilise the Choquet integral to scalarise multiple objectives, adaptively update the parameters using non–dominated solutions, which are obtained during the minimisation of scalarised sub–problems. Thanks to the non–linearity of Choquet integral, our method can capture non–convex, curved, disconnected and unevenly distributed Pareto front. The performance of CAS is demonstrated under different scenarios of ZDT1, ZDT2, ZDT3, DTLZ1, and DTLZ2. Besides, we benchmark CAS against a baseline algorithm, by replacing the Choquet integral by a linear scalarising function and updating parameters via `NNLS`. We claim that CAS offers slight improvement in terms of Hypervolume and number of non–dominated solutions found. Our proposed method and the experiments are publicly available in this repository.

Contents

Acknowledgements **2**

1 Introduction **7**

2 Related works **8**

3 Preliminaries **10**
3.1 Multi–objective Optimisation 10
3.2 Scalarisation method 11
3.3 Choquet capacities and integral 13

4 Choquet Adaptive Scalarisation (CAS) **17**
4.1 Motivation 17
4.2 Algorithm 17

5 Experiments **21**
5.1 Test problems 21
5.2 Performance metrics 22
5.3 Experimental setup 22
5.4 Results 23

6 Conclusion and discussion **29**

Appendix **34**

A Notations and Terminology **34**

Acknowledgements

I would like to express my sincere gratitude to A/Prof. Julien Ugon for his dedicated teaching of SIT316 - Linear and Nonlinear Optimisation, which furnished much of the theoretical grounding on which this thesis rests. Also, his inspiration and invaluable discussions throughout the trimester. Importantly, his constructive feedback on the drafts of this report and editorial suggestions.

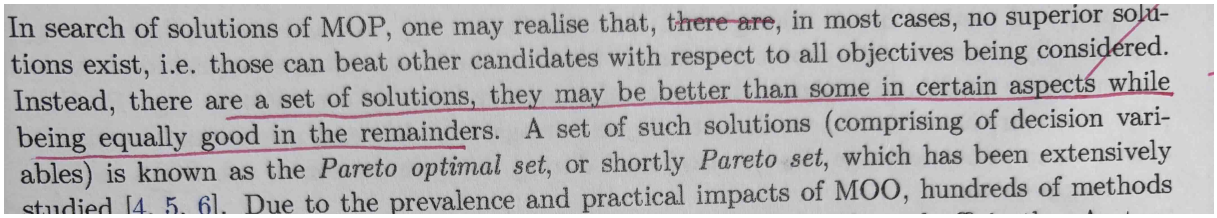
I am thankful for my supervisor, Prof. Gleb Beliakov, for proposing the direction of this work, providing the `ChoquetReg` routine upon which the capacity–learning step of this thesis is built, his well–established package `pyfmtools` for Choquet capacity estimation, and his patient guidance throughout its development.

The experiments reported herein were conducted with the `pyfmtools` and `pymoo` libraries. I thank their authors and maintainers for making this software freely available.

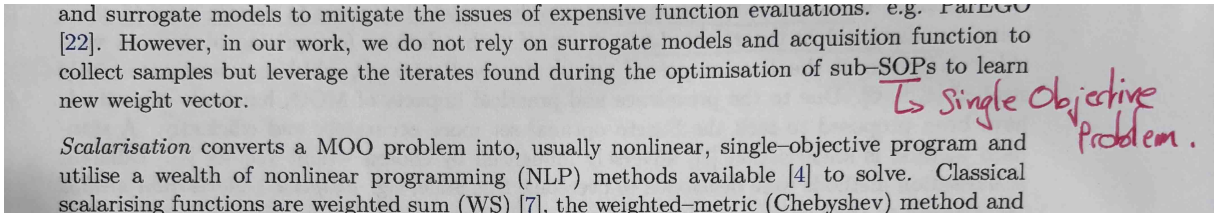
Finally, I thank my family and my friends for their unfailing support over the course of this trimester, and myself for always being consistent and curious in learning.

Response to feedback

- **Comment 1.** Wrong concepts given in **Sec. 1**. ⇒ Fix the underlined sentence and remove "there are".

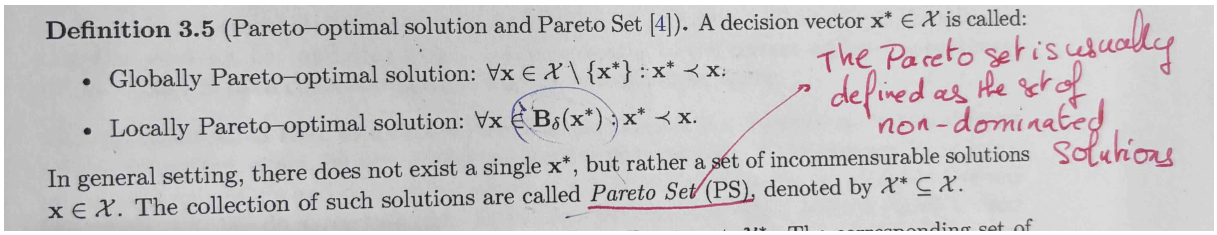

In search of solutions of MOP, one may realise that, there are, in most cases, no superior solutions exist, i.e. those can beat other candidates with respect to all objectives being considered. Instead, there are a set of solutions, they may be better than some in certain aspects while being equally good in the remainders. A set of such solutions (comprising of decision variables) is known as the *Pareto optimal set*, or shortly *Pareto set*, which has been extensively studied [4, 5, 6]. Due to the prevalence and practical impacts of MOO, hundreds of methods

- **Comment 2.** An acronym is not defined elsewhere earlier in **Sec. 2**. ⇒ Add its full name.


and surrogate models to mitigate the issues of expensive function evaluations. e.g. ParEGO [22]. However, in our work, we do not rely on surrogate models and acquisition function to collect samples but leverage the iterates found during the optimisation of sub–SOPs to learn new weight vector.

Scalarisation converts a MOO problem into, usually nonlinear, single–objective program and utilise a wealth of nonlinear programming (NLP) methods available [4] to solve. Classical scalarising functions are weighted sum (WS) [7], the weighted–metric (Chebyshev) method and

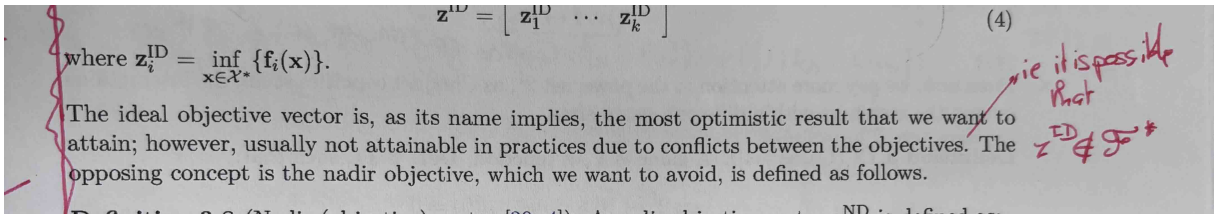
- **Comment 3.** In the **Sec. 3**:
  - Review **Def. 3.5**. ⇒ Within that definition, fix to non–dominated solution, rather than solutions dominating others. A short note about "local Pareto–optimallity".


Definition 3.5 (Pareto–optimal solution and Pareto Set [4]). A decision vector $\mathbf{x}^* \in \mathcal{X}$ is called:

- Globally Pareto–optimal solution: $\forall \mathbf{x} \in \mathcal{X} \setminus \{\mathbf{x}^*\} : \mathbf{x}^* \prec \mathbf{x}$.
- Locally Pareto–optimal solution: $\forall \mathbf{x} \in \mathbf{B}_\delta(\mathbf{x}^*) : \mathbf{x}^* \prec \mathbf{x}$.

In general setting, there does not exist a single $\mathbf{x}^*$, but rather a set of incommensurable solutions $\mathbf{x} \in \mathcal{X}$. The collection of such solutions are called *Pareto Set* (PS), denoted by $\mathcal{X}^* \subseteq \mathcal{X}$.

  - Under **Def. 3.7** and **Def. 3.8**, add a note that these points may not obtained in $\mathcal{F}^*$. ⇒ Add one sentence to clarify this, below **Def. 3.8**.


$$\mathbf{z}^{\text{ID}} = \begin{bmatrix} \mathbf{z}_1^{\text{ID}} & \cdots & \mathbf{z}_k^{\text{ID}} \end{bmatrix} \quad (4)$$

where $\mathbf{z}_i^{\text{ID}} = \inf_{\mathbf{x} \in \mathcal{X}^*} \{\mathbf{f}_i(\mathbf{x})\}$.

The ideal objective vector is, as its name implies, the most optimistic result that we want to attain; however, usually not attainable in practices due to conflicts between the objectives. The opposing concept is the nadir objective, which we want to avoid, is defined as follows.

  - Above **Sec. 3.3**, add another section introducing the weighted sum method and scalarisation method, in general. ⇒ Add a new section.

3.2 Choquet capacities and integral

We briefly discuss relevant notions to Choquet capacities and integral, an extensive research area in Fuzzy Measure theory. We confine ourselves to working with discrete sets, since the number of objectives in **Def. 3.1** is a positive integer. Nonetheless, the upcoming concepts can be generalised to infinite sets and/or continuous spaces, see [41, 42, 43]. To support the understanding of algorithms proposed later, we shall refer to the set below as the set of objectives

– Fix the set notations in **Sec. 3.3** ⇒ Define a new set $\mathcal{K}$ of indices of the objectives and align the definitions.

– Define "non–decreasing permutation" ⇒ Add **Def. 3.19**

– Fix the proof of **Remark. 3.21**.

– Modify **Def. 3.20**.

– Fix **Fig. 1**.

– Add Möbius transform and short note on capacity representations.

Definition 3.10 (Power set [41]). Let $\mathbf{f}$ be an arbitrary set, i.e. our collection of objectives. A power set is defined as:

$$2^{\mathbf{f}} \triangleq \{\mathbf{f}' \mid \mathbf{f}' \subseteq \mathbf{f}\}, \quad \mathbf{f} \in \mathbb{R}^k, \ \mathbf{f}' \in \mathbb{R}^{k'} \subset \mathbb{R}^k \tag{7}$$

Definition 3.11 (A set function [41]). A set function is given by:

$$\mu : \mathbf{f} \mapsto \mathbb{R}_{\geq 0} \tag{8}$$

From now, we pay more attention to the power set $2^{\mathbf{f}}$, as Choquet capcities study the interactions among the members within different group sizes.

Definition 3.12 (Game [44]). A game is a set function, **Def. 3.11**, such that:

$$\mu : 2^{\mathbf{f}} \mapsto \mathbb{R}_{\geq 0} \quad \text{and} \quad \mu(\emptyset) = 0 \tag{9}$$

Definition 3.13 (Premeasure [44]). A premeasure is a game, **Def. 3.12**, such that:

$$\mathcal{A} \subset \mathcal{B} \Rightarrow \mu(\mathcal{A}) \leq \mu(\mathcal{B}) \tag{10}$$

Definition 3.14 (Choquet Capacity [44]). The Choquet Capacity is a premeasure, by **Def. 3.13**. The Choquet capacity is given by:

$$\mu : 2^{\mathbf{f}} \mapsto \mathbb{R}_{\geq 0} \tag{11}$$

where $\mu(\emptyset) = 0$ and $\mathcal{A} \subset \mathcal{B} \subseteq 2^{\mathbf{f}} \Rightarrow \mu(\mathcal{A}) \leq \mu(\mathcal{B})$.

An intuitive way to interpret **Def. 3.14** is that, Choquet capacity represents the interactions between all possible combination of objectives in $\mathbf{f}$. Importantly, due to non–additivity, i.e. $\mu(\mathcal{A} \cup \mathcal{B}) \neq \mu(\mathcal{A}) + \mu(\mathcal{B})$, it possesses rich expressiveness, and thus, appropriately reflects the real–world interactions.

One may realise, every time we derive a new concept, we further impose more restrictions on the prior concept. We devote the following diagram as a summary for the above–mentioned concepts.

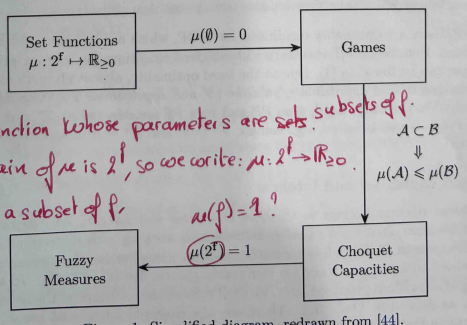
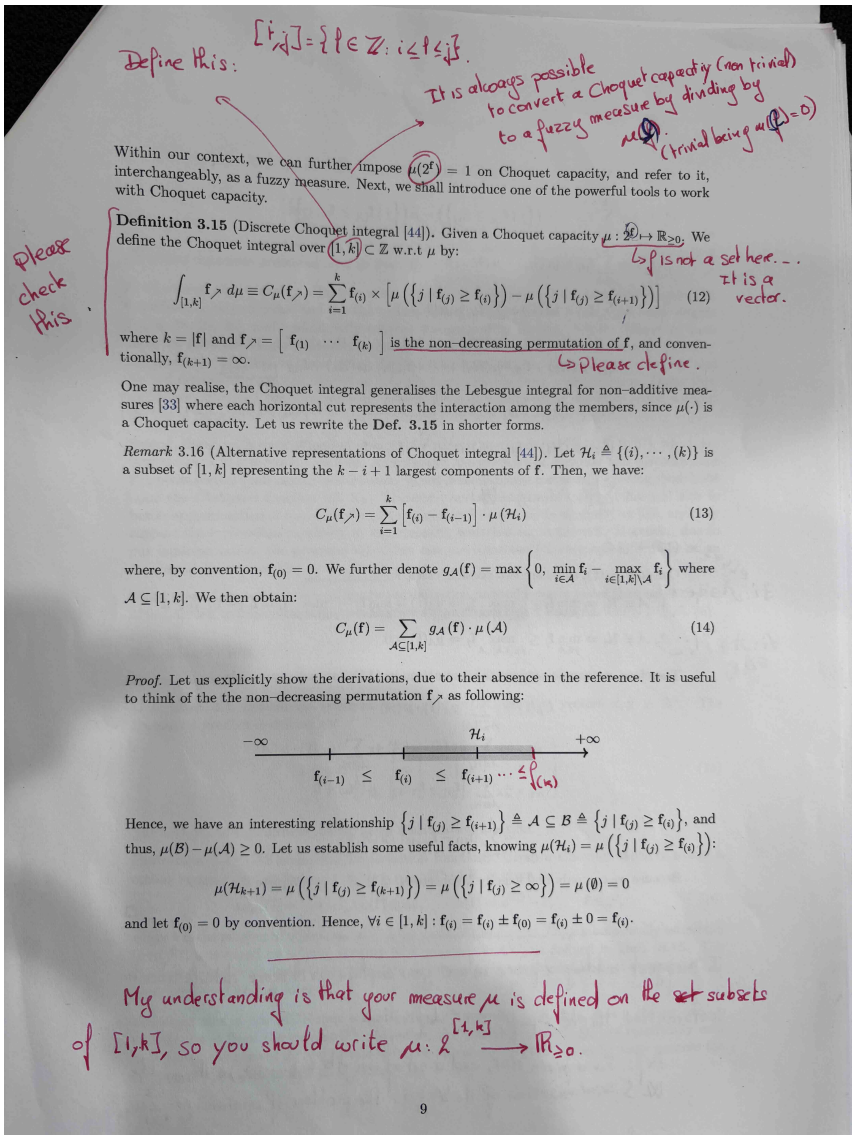


Figure 1: Simplified diagram, redrawn from [44].

8



Define this: $[i,j] = \{\ell \in \mathbb{Z} : i \le \ell \le j\}$.

It is always possible to convert a Choquet capacity (non trivial) to a fuzzy measure by dividing by $\mu(\Omega)$. (trivial being $\mu(\Omega)=0$)

Within our context, we can further impose $\mu(2^{\mathbf{f}}) = 1$ on Choquet capacity, and refer to it, interchangeably, as a fuzzy measure. Next, we shall introduce one of the powerful tools to work with Choquet capacity.

Please check this.

Definition 3.15 (Discrete Choquet integral [44]). Given a Choquet capacity $\mu : 2^{\mathbf{f}} \mapsto \mathbb{R}_{\ge 0}$. We define the Choquet integral over $[1,k] \subset \mathbb{Z}$ w.r.t μ by:

↳ $\mathbf{f}$ is not a set here... It is a vector.

$$\int_{[1,k]} \mathbf{f}_{\nearrow} \, d\mu \equiv C_\mu(\mathbf{f}_{\nearrow}) = \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \left[\mu\left(\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(i)}\}\right) - \mu\left(\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(i+1)}\}\right)\right] \quad (12)$$

where $k = |\mathbf{f}|$ and $\mathbf{f}_{\nearrow} = \begin{bmatrix} \mathbf{f}_{(1)} & \cdots & \mathbf{f}_{(k)} \end{bmatrix}$ is the non–decreasing permutation of $\mathbf{f}$, and conventionally, $\mathbf{f}_{(k+1)} = \infty$.

↳ Please define.

One may realise, the Choquet integral generalises the Lebesgue integral for non–additive measures [33] where each horizontal cut represents the interaction among the members, since $\mu(\cdot)$ is a Choquet capacity. Let us rewrite the **Def. 3.15** in shorter forms.

Remark 3.16 (Alternative representations of Choquet integral [44]). Let $\mathcal{H}_i \triangleq \{(i), \cdots, (k)\}$ is a subset of $[1,k]$ representing the $k-i+1$ largest components of $\mathbf{f}$. Then, we have:

$$C_\mu(\mathbf{f}_{\nearrow}) = \sum_{i=1}^{k} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu(\mathcal{H}_i) \quad (13)$$

where, by convention, $\mathbf{f}_{(0)} = 0$. We further denote $g_{\mathcal{A}}(\mathbf{f}) = \max\left\{0, \min_{i \in \mathcal{A}} \mathbf{f}_i - \max_{i \in [1,k] \setminus \mathcal{A}} \mathbf{f}_i\right\}$ where $\mathcal{A} \subseteq [1,k]$. We then obtain:

$$C_\mu(\mathbf{f}) = \sum_{\mathcal{A} \subseteq [1,k]} g_{\mathcal{A}}(\mathbf{f}) \cdot \mu(\mathcal{A}) \quad (14)$$

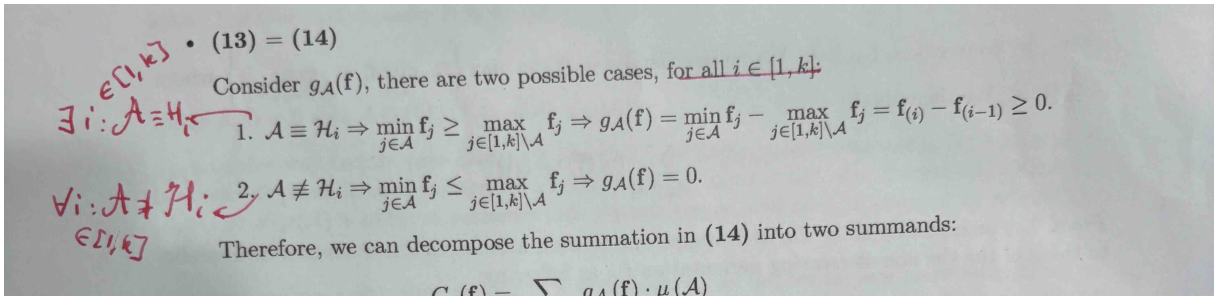
Proof. Let us explicitly show the derivations, due to their absence in the reference. It is useful to think of the the non–decreasing permutation $\mathbf{f}_{\nearrow}$ as following:

Hence, we have an interesting relationship $\left\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(i+1)}\right\} \triangleq \mathcal{A} \subseteq \mathcal{B} \triangleq \left\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(i)}\right\}$, and thus, $\mu(\mathcal{B}) - \mu(\mathcal{A}) \ge 0$. Let us establish some useful facts, knowing $\mu(\mathcal{H}_i) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(i)}\right\}\right)$:

$$\mu(\mathcal{H}_{k+1}) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \ge \mathbf{f}_{(k+1)}\right\}\right) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \ge \infty\right\}\right) = \mu(\emptyset) = 0$$

and let $\mathbf{f}_{(0)} = 0$ by convention. Hence, $\forall i \in [1,k] : \mathbf{f}_{(i)} = \mathbf{f}_{(i)} \pm \mathbf{f}_{(0)} = \mathbf{f}_{(i)} \pm 0 = \mathbf{f}_{(i)}$.

My understanding is that your measure μ is defined on the ~~set~~ subsets of $[1,k]$, so you should write $\mu : 2^{[1,k]} \longrightarrow \mathbb{R}_{\ge 0}$.



- **(13) = (14)**

$\exists i : \mathcal{A} = \mathcal{H}_i \in [1,k]$

Consider $g_{\mathcal{A}}(\mathbf{f})$, there are two possible cases, for all $i \in [1,k]$:

1. $\mathcal{A} \equiv \mathcal{H}_i \Rightarrow \min_{j \in \mathcal{A}} \mathbf{f}_j \ge \max_{j \in [1,k] \setminus \mathcal{A}} \mathbf{f}_j \Rightarrow g_{\mathcal{A}}(\mathbf{f}) = \min_{j \in \mathcal{A}} \mathbf{f}_j - \max_{j \in [1,k] \setminus \mathcal{A}} \mathbf{f}_j = \mathbf{f}_{(i)} - \mathbf{f}_{(i-1)} \ge 0.$

$\forall i : \mathcal{A} \ne \mathcal{H}_i \in [1,k]$

2. $\mathcal{A} \not\equiv \mathcal{H}_i \Rightarrow \min_{j \in \mathcal{A}} \mathbf{f}_j \le \max_{j \in [1,k] \setminus \mathcal{A}} \mathbf{f}_j \Rightarrow g_{\mathcal{A}}(\mathbf{f}) = 0.$

Therefore, we can decompose the summation in **(14)** into two summands:

$$C_\mu(\mathbf{f}) = \sum g_{\mathcal{A}}(\mathbf{f}) \cdot \mu(\mathcal{A})$$

- **Comment 4.** In **Sec. 4.2**:
  - Modify **Def. 4.3** and devoted a short note.
  - A short note on what "lexicographic order" is, below **Alg. 1**.
  - A short note on what FILTER$(\cdot)$ does, below **Alg. 1**.
  - Define what NLP$(\cdot)$ does, expanded **Alg. 1**.
  - Provide an informal sketch of how the algorithm conceptually works.

Definition 4.3 (In search of μ and $\mathbf{a}$).

$$\min_{\mathbf{a}\in\mathbb{R}^k} \arg\min_{\mu} \sum_{\mathbf{f}\in\hat{\mathcal{F}}^\star} [1 - k \cdot C_\mu(\mathbf{u}(\mathbf{f}) \odot \mathbf{a})]^2$$

We use the `Nelder-Mead` in `SciPy` [45] for the outer problem, and solve the inner one as a linear program with `linprog` (HiGHS) in `SciPy`.

Definition 4.4 (CAS sub–SOP). CAS solves the following sub–SOP at the t^{th} iteration:

$$\min_{\mathbf{x}\in\hat{\mathcal{X}}} s_{\mu^{(t)},\mathbf{a}^{(t)}}(\mathbf{f}(\mathbf{x})) \tag{17}$$

where $\mu, \mathbf{a}$ are found by solving problem defined in **Def. 4.3**.

Every $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$ is retained, and later filtered to obtain $\hat{\mathcal{X}}^\star$ and $\hat{\mathcal{F}}^\star$ at the t^{th} iteration.

As reviewed earlier in **Sec. 2**, two design choices distinguish **Alg. 1** from the adaptive scalarisation methods. First, exactly one sub–SOP is solved per iteration, rather than a series of them. Second, the routine NLP returns iterates $\mathcal{E}_t$ visited by the nonlinear solver before termination, which constitute the training data from which the next capacity is learnt. We offer the following diagram to demonstrate our algorithm workflow, i.e. **Alg. 1**.

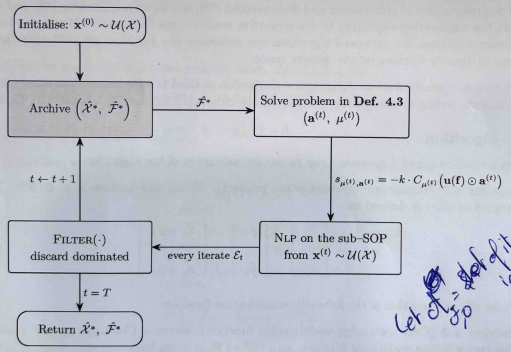


Figure 2: The flowchart of **Alg. 1**.

Algorithm 1 Choquet Adaptive Scalarisation (CAS)

Require: P_{MOP} over the box $\mathcal{X} = [\mathbf{x}^{\text{lb}}, \mathbf{x}^{\text{ub}}]$; budget T; tolerance τ; nonlinear solver NLP(·)
Ensure: Approximate Pareto set $\hat{\mathcal{X}}^\star$ and Pareto front $\hat{\mathcal{F}}^\star$

$\mathbf{x}^{(0)} \sim \mathcal{U}(\mathcal{X})$
$\mathcal{E}_0 \leftarrow \text{NLP}(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}), \mathbf{x}^{(0)}, \mathcal{X}, \tau)$ ▷ μ not yet fitted: $\mathbf{w} = \mathbf{1}/k$
$(\hat{\mathcal{X}}^\star, \hat{\mathcal{F}}^\star) \leftarrow \text{FILTER}(\mathcal{E}_0)$
for $t = 1, \ldots, T$ **do**
  Solve problem in **Def. 4.3** for $(\mathbf{a}^{(t)}, \mu^{(t)})$
  $\mathbf{x}^{(t)} \sim \mathcal{U}(\mathcal{X})$ ▷ random fresh start
  $\mathcal{E}_t \leftarrow \text{NLP}\left(s_{\mu^{(t)},\mathbf{a}^{(t)}}(\mathbf{f}(\mathbf{x})), \mathbf{x}^{(t)}, \mathcal{X}, \tau, \texttt{max_iter} = 50\right)$ ▷ all retained
  $(\hat{\mathcal{X}}^\star, \hat{\mathcal{F}}^\star) \leftarrow \text{FILTER}(\hat{\mathcal{F}}^\star \cup \mathcal{E}_t)$ ▷ discard dominated points
end for
return $\hat{\mathcal{X}}^\star$, $\hat{\mathcal{F}}^\star$

function FILTER($\mathcal{Z}$, *kind* = "strongly")
  $\mathcal{P} \leftarrow \emptyset$ ▷ $\leq, <$ component–wise; $\epsilon = 10^{-5}$
  if *kind* = "strongly" **then**
    for $\mathbf{z} \in \mathcal{Z}$ in lexicographic order **do** ▷ Faster filtering for strongly Pareto optimal
      if $\nexists\, \mathbf{z}' \in \mathcal{P} : \mathbf{z}' \leq \mathbf{z}$ **then**
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{z}\}$
      end if
    end for
  else if *kind* = "eps" **then**
    $\mathcal{P} \leftarrow \{\mathbf{z} \in \mathcal{Z} \mid \nexists\, \mathbf{z}' \in \text{FILTER}(\mathcal{Z}, \text{"strongly"}) : \mathbf{z}' < \mathbf{z} - \boldsymbol{\epsilon}\}$ ▷ one copy per $\mathbf{z}$
  else if *kind* = "weakly" **then**
    $\mathcal{P} \leftarrow \{\mathbf{z} \in \mathcal{Z} \mid \nexists\, \mathbf{z}' \in \mathcal{Z} \setminus \{\mathbf{z}\} : \mathbf{z}' < \mathbf{z}\}$
  end if
  return $\mathcal{P}$ and $\mathbf{f}^{-1}(\mathcal{P})$ ▷ $\mathbf{f}^{-1}(\mathcal{P})$ is already stored, not the inverse of $\mathbf{f}$
end function

We note here, that $\mathbf{f}^{-1}$ implies the pre–image for $\mathcal{P}$, and in fact, recorded at every iterate in **Alg. 1**. In other words, it is not an inverse function of $\mathbf{f}$, but a manipulation of notation. For sake of benchmarking, we shall propose a baseline algorithm, by replacing the scalarisation function of **Def. 4.2** by a linear combination $\mathbf{w}^\intercal \mathbf{f}(\mathbf{x})$, where $\mathbf{w}$ is updated by the current $\hat{\mathcal{F}}^\star$.

Algorithm 2 Baseline algorithm

Require: P_{MOP} over the box $\mathcal{X} = [\mathbf{x}^{\text{lb}}, \mathbf{x}^{\text{ub}}]$; budget T; tolerance τ; nonlinear solver NLP(·)
Ensure: Approximate Pareto set $\hat{\mathcal{X}}^\star$ and Pareto front $\hat{\mathcal{F}}^\star$

$\mathbf{x}^{(0)} \sim \mathcal{U}(\mathcal{X})$
$\mathbf{w} \leftarrow \mathbf{1}/k \in \mathbb{R}^k$
$\mathcal{E}_0 \leftarrow \text{NLP}(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}), \mathbf{x}^{(0)}, \mathcal{X}, \tau)$ ▷ uniform initial weights
$(\hat{\mathcal{X}}^\star, \hat{\mathcal{F}}^\star) \leftarrow \text{FILTER}(\mathcal{E}_0)$
for $t = 1, \ldots, T$ **do**
  $\mathbf{w} \leftarrow \arg\min_{\mathbf{w} \geq \mathbf{0}} \left\| \hat{\mathcal{F}}^\star \mathbf{w} - \mathbf{1} \right\|_2$ ▷ non–negative least squares
  $\mathbf{w} \leftarrow \mathbf{w} / \|\mathbf{w}\|_2$ if $\mathbf{w} \neq \mathbf{0}$, else $\mathbf{1}/k$ ▷ $\arg\min_{\mathbf{x}} \mathbf{w}^\intercal \mathbf{f}$ is scale–invariant
  $\mathbf{x}^{(t)} \sim \mathcal{U}(\mathcal{X})$ ▷ random fresh start
  $\mathcal{E}_t \leftarrow \text{NLP}\left(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}), \mathbf{x}^{(t)}, \mathcal{X}, \tau, \texttt{max_iter} = 50\right)$ ▷ all retained
  $(\hat{\mathcal{X}}^\star, \hat{\mathcal{F}}^\star) \leftarrow \text{FILTER}\left(\hat{\mathcal{F}}^\star \cup \mathcal{E}_t\right)$ ▷ discard dominated points
end for
return $\hat{\mathcal{X}}^\star$, $\hat{\mathcal{F}}^\star$

1 Introduction

Optimisation plays a pivotal role throughout the history. It may either appear explicitly as a problem–solving method or implicitly when completing daily tasks. A single-objective optimisation can be minimisation of time taken to complete a task or maximisation of profits in investment. Undoubtedly, single-objective optimisation is insufficient for complex problems, especially when we have to resolve the conflicts between the objectives, meaning improving some criteria resulting in worsening the others. Therefore, Multi–objective optimisation (MOO), or multicriteria optimisation or vector optimisation, is preferable and has been proven to yield better outcomes in real-world practices. The applications of MOO are, including but not limited to, balancing amongst conflicting objectives in smart cities [1], medical research community, specifically in drug design [2] and civil engineering sector, particularly in operational management of hydropower dams [3]. In the context of MOO, there are various concepts and issues associated with it. We shall confine ourselves solely to related concepts and avoid any unnecessary overcomplication. Furthermore, we shall assume that the Multi-objective Optimisation Problems (MOP) given herein is already mathematically well–defined.

In search of solutions of MOP, one may realise that, in most cases, no superior solutions exist, i.e. those can beat other candidates with respect to all objectives being considered. Instead, there are a set of solutions, they may be better than some in certain aspects while being worse in the remainders. A set of such solutions (comprising of decision variables) is known as the *Pareto optimal set*, or shortly *Pareto set*, which has been extensively studied [4, 5, 6]. Due to the prevalence and practical impacts of MOO, hundreds of methods have been proposed to seek the Pareto optimal set more accurately and efficiently. A standard method is scalarisation [7], adversely influenced by chosen weight vectors [4]. Different scalarisation methods were developed to overcome this issue, e.g. *adaptive scalarisation method* [8], *Chebyshev method* [4], and Normal Boundary Intersection (NBI) [9]. Nonetheless, it suffers from a critical disadvantage, without further assumptions, e.g. convexity, solution in scalarised problem may not correspond to any Pareto optimal solutions in the original problem [7, 10]. Modern family of algorithms is metaheuristics, e.g. Evolutionary–based techniques [11], Decomposition–based Evolutionary algorithm [12] and Multi–objective Ant Colony Optimisation [13]. Another recently emergent approach is known as Active Learning (AL)–based MOO, which can be used to accelerate the evolutionary-based MOO algorithms [14]. Also, it is adopted as more informed sampling techniques, rather than exhaustive search for the Pareto optimal set, e.g. Pareto(–guided) Active Learning (PAL) [10] and ϵ–PAL [15]. Another powerful yet confused with PAL is Bayesian Optimisation (BO), useful when when neither closed-form, well-behaved objectives are available nor function evaluations are inexpensive [16].

Our contributions in this thesis are:

- A proposed algorithm, Choquet Adaptive Scalarisation (CAS).
- An empirical benchmark of CAS against other MOP solvers and a linear version of our framework.

We structure the remainder of the thesis as follows. In **Sect. 2**, we review seemingly related work in MOO, recent developments, and highlight differences between their and our method. **Sect. 3** presents important concepts in MOO and a brief introduction to Choquet integral. **Sect. 4** proposes our algorithm and shows a theoretical analysis of the method. In **Sect. 5**, we conduct an empirical experiment to benchmark our proposed technique. Eventually, we close the thesis in **Sect. 6** by delivering our insights and possible future works. Readers may refer to **Appendix** for notations used and related topics.

2 Related works

In this section, we summarise findings in certain papers that are seemingly related to this thesis and emphasise the differences between our method and theirs.

Active Learning, also known as query learning and optimal experimental design, is a research area in Artificial Intelligence (AI) community. The governing idea is that, active learners intelligently select and learn from data that are deemed as representative and informative, while maintaining a comparative performance with those learnt from a more extensive volume of data [17]. It has been widely applied to various tasks in AI community, e.g. classification tasks [18], speech recognition tasks [19] and Computer Vision [20]. Therefore, it aims to approximate the true unknown function on the whole domain. Meanwhile, *Bayesian Optimisation* is a subset of AL, as illustrated in [21], whose aim is to seek an optimal solution of the objective function. They are essentially useful when the objective functions are computationally prohibitive to evaluate, and both of them rely on the core idea of *adaptive sampling*. Due to the practical assumption regarding objective functions in AL paradigm, its framework is widely adopted to solve MOO problems. In either paradigm, scalarisation of multi–objectives are optional, which differ from our proposed method herein, though still share similarity, e.g. adaptive sampling strategy. Additionally, these methods choose new sample based on the construction of acquisition function and surrogate models to mitigate the issues of expensive function evaluations. e.g. ParEGO [22]. However, in our work, we do not rely on surrogate models and acquisition function to collect samples but leverage the iterates found during the optimisation of sub single–objective problems (sub–SOPs) to learn new weight vector.

Scalarisation converts a MOO problem into, usually nonlinear, single–objective program and utilise a wealth of nonlinear programming (NLP) methods available [4] to solve. Classical scalarising functions are weighted sum (WS) [7], the weighted–metric (Chebyshev) method and ϵ–constraint method [23]. A critical limitation of scalarisation method, is the solution attained in scalarised problem may not correspond to the Pareto optimal solutions in the original problem, e.g. when the Pareto front is non–convex [4, 9]. Besides, some disadvantages resemble prevalent issues in regression analysis. One issue is multicollinearity [24], meaning that some objectives may be a functional representation of one or many other objectives presented in the problem, and thus adversely influence our weight learning process. Another obstacle is difference in units of objectives, e.g. some objectives can be excessively large, says over 1000, whereas some are bounded between 0 and 1. Directly learning may amount to bias towards objectives with higher units, and thus, normalisation methods in evolutionary algorithms for MOO were studied in [25, 26].

Therefore, *adaptive scalarisation* was devised to overcome some issues above. The adaptive weighted–sum method [27] refines weights only in regions that remain sparsely sampled and imposes additional inequality constraints so that non–convex regions become attainable. The same principle recurs in decomposition–based evolutionary algorithms as weight–vector adaptation towards under–represented regions of irregular PFs [28, 29, 12]. Our method adopts the adaptive sampling strategy, i.e. updating weight vector according to the current explored approximate PF, but differ in data collection methodology to update weight vector.

Decomposition–based refer to a class of methods partitioning the objective space into different regions to explore, again, by use of *adaptive scalarisation* methods. It decomposes P_{MOP} into a sequence of sub–SOPs and optimises a subset of sub–SOPs concurrently in a collaborative way [30, 29, 28, 12]. We do agree with them in terms of decomposing into a series of sub–SOPs and adaptive scalarisation, but instead of partitioning of the objective space, we iteratively learn from the current approximate PF in the objective space.

Multi–objective Optimistic Search (MOOS), a subset of *Multi–objective Design Space Exploration*

framework [31]. A notable work that is closely related to our algorithmic design is given in [32]. We share a quite similar idea of local search for approximate PF, but significantly diverge from the choice of scalarisation function, termination criteria and hierarchical partition of decision space while seeking approximate PS/PF.

Therefore, our method adopts partially the design of above–mentioned frameworks and adopts the discrete Choquet integral [33] as a scalarisation function. We believe Choquet integral capable of approximating non–convex PF as it assigns weights to a subset of objectives via the use of a fuzzy measure [33], which employs nonlinearity. It has been employed as a decision maker's preference model to rank candidates in interactive evolutionary MOO [34] and for multicriteria decision–making purposes [35, 36]. However, to the best of our knowledge, not yet been leveraged in an adaptive scalarisation paradigm. This is the gap the present thesis addresses.

3 Preliminaries

3.1 Multi–objective Optimisation

Throughout this report, we aim to solve the MOP defined as below.

Definition 3.1 (Multi–objective Optimisation Problem (MOP) [4])**.**

$$P_{\text{MOP}} \triangleq \min_{\mathbf{x}\in\mathcal{X}} \mathbf{f}(\mathbf{x}) \tag{1}$$

where $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix}$, $\mathcal{X} \subset \mathbb{R}^n$ is a compact set and $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^k$ given by $\mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{f}_1(\mathbf{x}) & \cdots & \mathbf{f}_k(\mathbf{x}) \end{bmatrix}$ with $\mathbf{f}_i : \mathbb{R}^n \mapsto \mathbb{R}$.

Due to the definition of $\mathbf{f}$, MOP can be interchangeably referred to as a vector optimisation problem. Also, some literatures use many–objective optimisation to emphasise the presence of more than three objectives [2, 26, 37]. For sake of generality, we shall consider P_{MOP} a nonlinear optimisation problem, unless stated otherwise. In addition, we call $\mathcal{X}$ a *feasible decision/design region/space*, and $\mathbf{x} \in \mathcal{X}$ are *decision vectors*. Likewise, the following set:

$$\mathcal{F} \triangleq \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\} \subseteq \mathbb{R}^k \tag{2}$$

is called a *feasible objective region/space*, and any $\mathbf{z} \in \mathcal{F}$ is an *objective vector*.

In most real-world scenarios, P_{MOP} is ill–posed [38], i.e. increments in some $\mathbf{f}_i$ result in decrements in other $\mathbf{f}_j$. Therefore, in MOO, we mostly refer to *Pareto optimality*. Similar to the single–objective optimisation problem (SOP), there exists global and local solutions. Denote $\mathbf{B}_\delta(\mathbf{x}) \triangleq \{\mathbf{y} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{y}\| < \delta\}$ as an (open) neighbourhood of $\mathbf{x} \in \mathcal{X}$. Let us discuss some relevant notions.

Definition 3.2 (Strong Pareto-dominance [37, 39])**.** Given $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Then, $\mathbf{x}$ strongly Pareto dominates $\mathbf{y}$, or $\mathbf{x} \prec \mathbf{y}$, iff $\forall i \in \{1, \ldots, k\} : \mathbf{f}_i(\mathbf{x}) \leq \mathbf{f}_i(\mathbf{y})$ and $\exists j \in \{1, \ldots, k\} : \mathbf{f}_j(\mathbf{x}) < \mathbf{f}_j(\mathbf{y})$.

Definition 3.3 ((Strong) ϵ–Pareto-dominance [15])**.** Given $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Then, $\mathbf{x}$ ϵ–Pareto dominates $\mathbf{y}$, or $\mathbf{x} \prec_{\boldsymbol{\epsilon}} \mathbf{y}$, iff $\forall i \in \{1, \ldots, k\} : \mathbf{f}_i(\mathbf{x}) < \mathbf{f}_i(\mathbf{y}) - \boldsymbol{\epsilon}_i$, for $\boldsymbol{\epsilon} \in \mathbb{R}^k$.

Definition 3.4 (Weak Pareto-dominance [4])**.** Given $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Then, $\mathbf{x}$ weakly Pareto dominates $\mathbf{y}$, or $\mathbf{x} \preceq \mathbf{y}$, iff $\forall i \in \{1, \ldots, k\} : \mathbf{f}_i(\mathbf{x}) \leq \mathbf{f}_i(\mathbf{y})$.

For sake of implementation, we can alternatively distinguish weakly and strongly Pareto dominance, given $\mathbf{x} \in \mathcal{X}$, it is NOT:

- Strongly Pareto optimal: $\exists \mathbf{y} \in \mathcal{X} : \mathbf{f}_i(\mathbf{y}) \leq \mathbf{f}_i(\mathbf{x}), \forall i \in \{1, \ldots, k\}$
- Strongly ϵ–Pareto optimal: $\exists \mathbf{y} \in \mathcal{X} : \mathbf{f}_i(\mathbf{y}) - \boldsymbol{\epsilon}_i < \mathbf{f}_i(\mathbf{x}), \forall i \in \{1, \ldots, k\}$
- Weakly Pareto optimal: $\exists \mathbf{y} \in \mathcal{X} : \mathbf{f}_i(\mathbf{y}) < \mathbf{f}_i(\mathbf{x}), \forall i \in \{1, \ldots, k\}$.

Definition 3.5 (Pareto–optimal solution and Pareto Set [4])**.** A decision vector $\mathbf{x}^* \in \mathcal{X}$ is called:

- Globally Pareto–optimal solution: $\nexists \mathbf{x} \in \mathcal{X} : \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.
- Locally Pareto–optimal solution: $\exists \delta > 0, \nexists \mathbf{x} \in \mathbf{B}_\delta(\mathbf{x}^*) : \mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

In general setting, there does not exist a single $\mathbf{x}^*$, but rather a set of incommensurable solutions $\mathbf{x} \in \mathcal{X}$. The collection of such solutions are called *Pareto Set* (PS), denoted by $\mathcal{X}^* \subseteq \mathcal{X}$. We further note here, the locally Pareto–optimal solution refer to the neighbourhood around itself, not taking into account all solutions to define the neighbourhood.

Definition 3.6 (Pareto front(ier) [4])**.** Consider the Pareto set $\mathcal{X}^*$. The corresponding set of objective vectors:

$$\mathcal{F}^* \triangleq \{\mathbf{f}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{X}^*\} \subseteq \mathcal{F} \tag{3}$$

is called the *Pareto Front* (PF), and each $\mathbf{z} \in \mathcal{F}^*$ is known as a *Pareto frontier*.

Likewise, we cannot guarantee the solution found by any existing MOO algorithms is globally Pareto optimal, unless special properties are given, e.g. convexity. Hence, for brevity, we refer to local Pareto–optimality as Pareto–optimality throughout this report.

Definition 3.7 (Ideal (objective) vector [39, 4])**.** An ideal objective vector $\mathbf{z}^{\text{ID}}$ is defined as:

$$\mathbf{z}^{\text{ID}} = \begin{bmatrix} \mathbf{z}_1^{\text{ID}} & \cdots & \mathbf{z}_k^{\text{ID}} \end{bmatrix} \tag{4}$$

where $\mathbf{z}_i^{\text{ID}} = \inf_{\mathbf{x} \in \mathcal{X}^*} \{\mathbf{f}_i(\mathbf{x})\}$.

The ideal objective vector is, as its name implies, the most optimistic result that we want to attain; however, usually not attainable in practices due to conflicts between the objectives. The opposing concept is the nadir objective, which we want to avoid, is defined as follows.

Definition 3.8 (Nadir (objective) vector [39, 4])**.** A nadir objective vector $\mathbf{z}^{\text{ND}}$ is defined as:

$$\mathbf{z}^{\text{ND}} = \begin{bmatrix} \mathbf{z}_1^{\text{ND}} & \cdots & \mathbf{z}_k^{\text{ND}} \end{bmatrix} \tag{5}$$

where $\mathbf{z}_i^{\text{ND}} = \sup_{\mathbf{x} \in \mathcal{X}^*} \{\mathbf{f}_i(\mathbf{x})\}$.

Generally speaking, $\mathbf{z}^{\text{ID}} \notin \mathcal{F}^*$ and $\mathbf{z}^{\text{ND}} \notin \mathcal{F}^*$. Unlike ideal objective vector, which is easier to find, simply by seeking minimiser of $\mathbf{f}$ component–wise on $\mathcal{X}$, which automatically arrives at $\mathbf{x} \in \mathcal{X}$. Finding maximiser of $\mathbf{f}$ component–wise on $\mathcal{X}$ may result in $\mathbf{x} \notin \mathcal{X}$, which contradicts our definition, hence harder. In practices, to avoid computing $\mathbf{z}^{\text{ID}}$ and $\mathbf{z}^{\text{ND}}$, one can use reference objective vectors found so far during the algorithm.

Definition 3.9 (Reference ideal and nadir objective vector)**.** Consider all solutions found by the i^{th} algorithm $\mathcal{A}^{(i)}$ as $\mathcal{A}_{\mathcal{F}}^{(i)} \triangleq \left\{\mathbf{f}^k\right\}_{k=1}^{K}$ after few iterations. Let $\mathcal{A}_{\mathcal{F}}^{\cup} = \cup_i \mathcal{A}_{\mathcal{F}}^{(i)}$ be the set of all solutions of all algorithms being considered. Then:

$$\mathbf{z}^{\text{ID}} \approx \mathbf{z}_{\text{ref}}^{\text{ID}} = \inf_{\mathbf{z} \in \mathcal{A}_{\mathcal{F}}^{\cup}} \mathbf{z} \qquad \text{and} \qquad \mathbf{z}^{\text{ND}} \approx \mathbf{z}_{\text{ref}}^{\text{ND}} = \sup_{\mathbf{z} \in \mathcal{A}_{\mathcal{F}}^{\cup}} \mathbf{z} \tag{6}$$

Nonetheless, these approximations are not good in early iterations of $\mathcal{A}$ [37], and may deteriorate performance of certain classes of algorithms solving MOP that rely on the knowledge of $\mathbf{z}^{\text{ID}}$ and $\mathbf{z}^{\text{ND}}$. Therefore, better yet advanced techniques are discussed in [40].

Similar to SOP, there are optimality conditions for MOP, which are known as the Pareto optimality conditions. Details of KKT necessary and sufficient conditions for Pareto Optimality, and related concepts can be found in [4]. Due to the local optimality, almost all methods proposing different approaches to find *approximate/putative PF* and *approximate PS*, denoted as $\hat{\mathcal{F}}^*$ and $\hat{\mathcal{X}}^*$, respectively. On the other hand, *true PF* and *true PS* are denoted as $\mathcal{F}^*$ and $\mathcal{X}^*$, respectively. When P_{MOP} is well-behaved, e.g. convex, $\hat{\mathcal{F}}^* = \mathcal{F}$ and $\hat{\mathcal{X}}^* = \mathcal{X}$ supposing local PF and PS can be found.

3.2 Scalarisation method

In short, the idea governing this method is, there are no unique gradient directions in MOP, since the objectives possibly conflict, and heading towards the gradient of some objectives may

result in the opposite gradient direction of others, and vice versa. Thus, it aims to transform the vector–valued objective into a scalar–valued objective and immediately attain the corresponding gradient direction, which may offer insightful guide to seek Pareto–optimal solutions. Regardless of whichever scalarisation functions or weight updating schemes are used, the ultimate goal of this method is to solve the following problem.

Definition 3.10 (Scalarisation method)**.** To determine the solutions to **Def. 3.1**, scalarisation method alternatively obtains solutions by solving the following problem.

$$\min_{\mathbf{x}\in\mathcal{X}} \; s(\mathbf{f}) \tag{7}$$

where $s : R^k \mapsto \mathbb{R}$ and $\mathbf{f} : \mathcal{X} \mapsto \mathbb{R}^k$.

Next, we shall review some scalarising functions used in this class of method. The simplest and most popular family of scalarising function is the weighted L_p scalarising method.

Definition 3.11 (Weighted L_p scalarising/scalarisation function [37])**.** Given the MOP as defined in **Def. 3.1**. Its multi–objective vector can be scalarised by:

$$s_{\mathbf{w},p}(\mathbf{f}) = \left(\sum_{i=1}^{k} \mathbf{w}_i \left|\mathbf{f}_i - \mathbf{z}^{\mathrm{ID}}\right|^p\right)^{\frac{1}{p}} \tag{8}$$

where $\mathbf{w} \in \mathbb{R}^k_{\geq 0} : \sum_i \mathbf{w}_i = 1$ is a weighting vector.

Therefore, by varying p, we obtain different scalarisation functions, and two most commonly used are Weighted Sum (WS) method and Chebyshev method.

Definition 3.12 (WS and Chebyshev methods [37])**.** The two methods are given by:

- WS method ($p = 1$):

$$s^{\mathrm{ws}}_{\mathbf{w}}(\mathbf{f}) = \left|\mathbf{w}^{\intercal}\left(\mathbf{f} - \mathbf{z}^{\mathrm{ID}}\right)\right| \tag{9}$$

- Chebyshev method ($p = \infty$):

$$s^{\mathrm{ch}}_{\mathbf{w}}(\mathbf{f}) = \max_{i=1}^{k}\left\{\mathbf{w}_i \left|\mathbf{f}_i - \mathbf{z}^{\mathrm{ID}}_i\right|\right\} \tag{10}$$

Remarkably, this method aims to close the gap between the current Pareto front and the ideal point, and thereby, signs of $\mathbf{f}$ are not our concerns. By altering $\mathbf{w}$, one can obtain a better or worse $\hat{\mathcal{F}}^*$, or differently, $\mathbf{w}$ can be adaptively updated from $\hat{\mathcal{F}}^*$ found so far.

Another scalarising function is also used in the context of MOO is Penalty–based scalarisation function (PBI).

Definition 3.13 (Penalty–based scalarisation/scalarising function [41])**.** Given the MOP as defined in **Def. 3.1**. Its multi–objective vector can be scalarised by:

$$s^{\mathrm{pbi}}_{\mathbf{w}}(\mathbf{f}) = d_1 + \theta d_2 \tag{11}$$

Let $\|\cdot\|$ be the Euclidean norm, then:

$$d_1 = \frac{\left\|\mathbf{w}^{\intercal}\left(\mathbf{f} - \mathbf{z}^{\mathrm{ID}}\right)\right\|}{\|\mathbf{w}\|} \qquad d_2 = \left\|\mathbf{f} - \left(\mathbf{z}^{\mathrm{ID}} + d_1\frac{\mathbf{w}}{\|\mathbf{w}\|}\right)\right\|$$

3.3 Choquet capacities and integral

We briefly discuss relevant notions to Choquet capacities and integral, an extensive research area in Fuzzy Measure theory. We confine ourselves to working with discrete sets, since the number of objectives in **Def. 3.1** is a positive integer. Nonetheless, the upcoming concepts can be generalised to infinite sets and/or continuous spaces, see [42, 43, 44]. To support the understanding of algorithms proposed later, we shall refer to the set below as the set of objectives in our MOPs, as defined in **Def. 3.1**. Therefore, the concepts introduced below, are not as general as found in their corresponding references.

Let $\mathcal{K} \triangleq \{1, \cdots, k\}$ be the set of indices of the objectives, $\mathbb{R}_{\geq 0} \triangleq [0, +\infty)$. We shall define the following definitions. We shall pay more attention to the power set $2^{\mathcal{K}}$, as Choquet capacities study the interactions among the members within different group sizes.

Definition 3.14 (Power set [42])**.** A power set is defined as:

$$2^{\mathcal{K}} \triangleq \{\mathcal{K}' \mid \mathcal{K}' \subseteq \mathcal{K}\} \tag{12}$$

Definition 3.15 (A set function [42])**.** A set function is given by:

$$\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0} \tag{13}$$

Definition 3.16 (Game [45])**.** A game is a set function, **Def. 3.15**, such that:

$$\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0} \quad \text{and} \quad \mu(\emptyset) = 0 \tag{14}$$

Definition 3.17 (Premeasure [45])**.** A premeasure is a game, **Def. 3.16**, such that:

$$\mathcal{A} \subset \mathcal{B} \Rightarrow \mu(\mathcal{A}) \leq \mu(\mathcal{B}) \tag{15}$$

Definition 3.18 (Choquet Capacity [45])**.** The Choquet Capacity is a premeasure, by **Def. 3.17**. The Choquet capacity is given by:

$$\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0} \tag{16}$$

where $\mu(\emptyset) = 0$ and $\mathcal{A} \subset \mathcal{B} \subseteq \mathcal{K} \Rightarrow \mu(\mathcal{A}) \leq \mu(\mathcal{B})$.

An intuitive way to interpret **Def. 3.18** is that, Choquet capacity represents the interactions between all possible combination of objectives in $\mathbf{f}$. Importantly, due to non–additivity, i.e. $\mu(\mathcal{A} \cup \mathcal{B}) \neq \mu(\mathcal{A}) + \mu(\mathcal{B})$, it possesses rich expressiveness, and thus, appropriately reflects the real–world interactions.

One may realise, every time we derive a new concept, we further impose more restrictions on the prior concept. We devote the following diagram as a summary for the above–mentioned concepts.

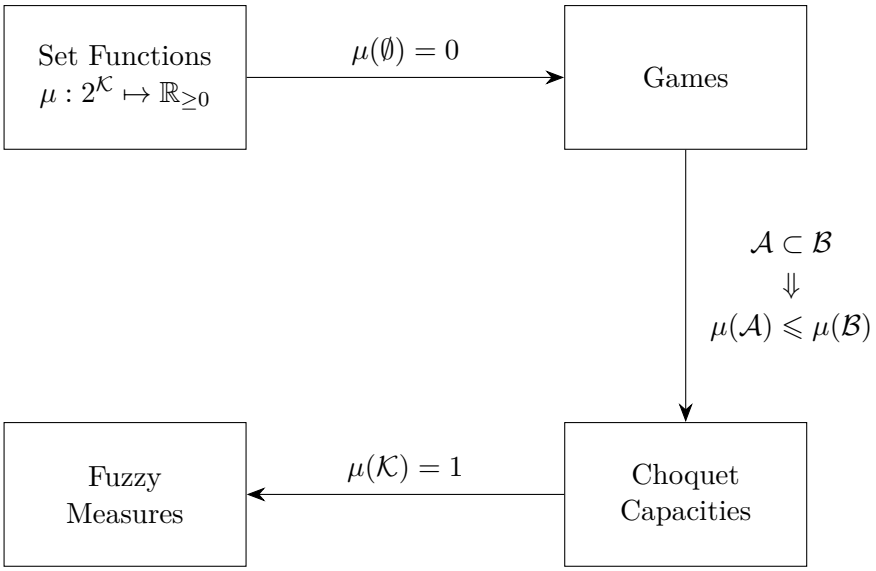


Figure 1: Simplified diagram, redrawn from [45].

Within our context, we can further impose $\mu(\mathcal{K}) = 1$ on Choquet capacity, and refer to it, interchangeably, as a fuzzy measure. Next, we shall introduce one of the powerful tools to work with Choquet capacity.

Definition 3.19 (Non–decreasing permutation [45])**.** Let $\mathbf{f}_{\nearrow}$ be a non–decreasing permutation of $\mathbf{f}$, meaning its components are rearranged, such that:

$$\mathbf{f}_{(1)} \leq \mathbf{f}_{(2)} \leq \cdots \leq \mathbf{f}_{(k-1)} \leq \mathbf{f}_{(k)} \tag{17}$$

Definition 3.20 (Discrete Choquet integral [45])**.** Given a Choquet capacity $\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0}$. We define the Choquet integral over $\mathcal{K}$ w.r.t μ by:

$$C_{\mu}(\mathbf{f}_{\nearrow}) = \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \left[\mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i)}\right\}\right) - \mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i+1)}\right\}\right)\right] \tag{18}$$

where $k = |\mathbf{f}|$ and $\mathbf{f}_{\nearrow} = \begin{bmatrix} \mathbf{f}_{(1)} & \cdots & \mathbf{f}_{(k)} \end{bmatrix}$ is the non–decreasing permutation of $\mathbf{f}$, and conventionally, $\mathbf{f}_{(k+1)} = \infty$.

One may realise, the Choquet integral generalises the Lebesgue integral for non–additive measures [33] where each horizontal cut represents the interaction among the members, since $\mu(\cdot)$ is a Choquet capacity. Let us rewrite the **Def. 3.20** in shorter forms.

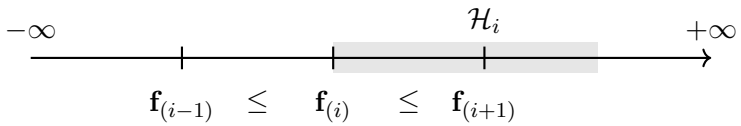
Remark 3.21 (Alternative representations of Choquet integral [45]). Let $\mathcal{H}_i \triangleq \{(i), \cdots, (k)\}$ is a subset of $\mathcal{K}$ representing the $k - i + 1$ largest components of $\mathbf{f}$. Then, we have:

$$C_{\mu}(\mathbf{f}_{\nearrow}) = \sum_{i=1}^{k} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu\left(\mathcal{H}_i\right) \tag{19}$$

where, by convention, $\mathbf{f}_{(0)} = 0$. We further denote $g_{\mathcal{A}}(\mathbf{f}) = \max\left\{0, \min_{i \in \mathcal{A}} \mathbf{f}_i - \max_{i \in \mathcal{K} \setminus \mathcal{A}} \mathbf{f}_i\right\}$ where $\mathcal{A} \subseteq \mathcal{K}$. We then obtain:

$$C_{\mu}(\mathbf{f}) = \sum_{\mathcal{A} \subseteq \mathcal{K}} g_{\mathcal{A}}\left(\mathbf{f}\right) \cdot \mu\left(\mathcal{A}\right) \tag{20}$$

Proof. Let us explicitly show the derivations, due to their absence in the reference. It is useful to think of the the non–decreasing permutation $\mathbf{f}_{\nearrow}$ as following:



Hence, we have an interesting relationship $\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i+1)}\right\} \triangleq \mathcal{A} \subseteq \mathcal{B} \triangleq \left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i)}\right\}$, and thus, $\mu(\mathcal{B}) - \mu(\mathcal{A}) \geq 0$. Let us establish some useful facts, knowing $\mu(\mathcal{H}_i) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i)}\right\}\right)$:

$$\mu(\mathcal{H}_{k+1}) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(k+1)}\right\}\right) = \mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \infty\right\}\right) = \mu\left(\emptyset\right) = 0$$

and let $\mathbf{f}_{(0)} = 0$ by convention. Hence, $\forall i \in \mathcal{K} : \mathbf{f}_{(i)} = \mathbf{f}_{(i)} \pm \mathbf{f}_{(0)} = \mathbf{f}_{(i)} \pm 0 = \mathbf{f}_{(i)}$.

- **(18) = (19)**

$$\begin{aligned}
C_\mu(\mathbf{f}_{\nearrow}) &= \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \left[\mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i)}\right\}\right) - \mu\left(\left\{j \mid \mathbf{f}_{(j)} \geq \mathbf{f}_{(i+1)}\right\}\right)\right] \\
&= \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \left[\mu\left(\mathcal{H}_i\right) - \mu\left(\mathcal{H}_{i+1}\right)\right] \\
&= \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \mu\left(\mathcal{H}_i\right) - \sum_{i=1}^{k} \mathbf{f}_{(i)} \times \mu\left(\mathcal{H}_{i+1}\right) \\
&= \mathbf{f}_{(1)} \cdot \mu\left(\mathcal{H}_1\right) + \mathbf{f}_{(2)} \cdot \mu\left(H_2\right) + \cdots + \mathbf{f}_{(k-1)} \cdot \mu\left(H_{k-1}\right) + \mathbf{f}_{(k)} \cdot \mu\left(\mathcal{H}_k\right) \\
&- \mathbf{f}_{(1)} \cdot \mu\left(\mathcal{H}_2\right) + \mathbf{f}_{(2)} \cdot \mu\left(\mathcal{H}_3\right) + \cdots + \mathbf{f}_{(k-1)} \cdot \mu\left(\mathcal{H}_k\right) + \mathbf{f}_{(k)} \cdot \underbrace{\mu\left(\mathcal{H}_{k+1}\right)}_{=0} \\
&= \mathbf{f}_{(1)} \cdot \mu\left(\mathcal{H}_1\right) + \left[\mathbf{f}_{(2)} \cdot \mu\left(\mathcal{H}_2\right) - \mathbf{f}_{(1)} \cdot \mu\left(\mathcal{H}_2\right)\right] + \cdots + \left[\mathbf{f}_{(k)} \cdot \mu\left(\mathcal{H}_k\right) - \mathbf{f}_{(k-1)} \cdot \mu\left(\mathcal{H}_k\right)\right] \\
&= \left[\mathbf{f}_{(1)} \cdot \mu\left(\mathcal{H}_1\right) - \mathbf{f}_{(0)} \cdot \mu\left(\mathcal{H}_1\right)\right] + \sum_{i=2}^{k} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu(\mathcal{H}_i) \\
&= \sum_{i=1}^{k} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu(\mathcal{H}_i)
\end{aligned}$$

- **(19) = (20)**

  Consider $g_{\mathcal{A}}(\mathbf{f})$, $\forall \mathcal{A} \subseteq \mathcal{K}$, there are two possible cases:

  1. $\exists i \in \mathcal{K} : \mathcal{A} \equiv \mathcal{H}_i \Rightarrow \min\limits_{j \in \mathcal{A}} \mathbf{f}_j \geq \max\limits_{j \in \mathcal{K} \setminus \mathcal{A}} \mathbf{f}_j \Rightarrow g_{\mathcal{A}}(\mathbf{f}) = \min\limits_{j \in \mathcal{A}} \mathbf{f}_j - \max\limits_{j \in \mathcal{K} \setminus \mathcal{A}} \mathbf{f}_j = \mathbf{f}_{(i)} - \mathbf{f}_{(i-1)} \geq 0.$
  2. $\forall i \in \mathcal{K} : \mathcal{A} \not\equiv \mathcal{H}_i \Rightarrow \min\limits_{j \in \mathcal{A}} \mathbf{f}_j \leq \max\limits_{j \in \mathcal{K} \setminus \mathcal{A}} \mathbf{f}_j \Rightarrow g_{\mathcal{A}}(\mathbf{f}) = 0.$

Therefore, we can decompose the summation in **(20)** into two summands:

$$\begin{aligned} C_\mu(\mathbf{f}) &= \sum_{\mathcal{A}\subseteq\mathcal{K}} g_{\mathcal{A}}(\mathbf{f}) \cdot \mu(\mathcal{A}) \\ &= \sum_{\mathcal{A}\equiv\mathcal{H}_i} g_{\mathcal{A}}(\mathbf{f}) \cdot \mu(\mathcal{A}) + \sum_{\mathcal{A}\not\equiv\mathcal{H}_i} g_{\mathcal{A}}(\mathbf{f}) \cdot \mu(\mathcal{A}) \\ &= \sum_{\mathcal{A}\equiv\mathcal{H}_i} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu(\mathcal{H}_i) + 0 \\ &= \sum_{i=1}^{k} \left[\mathbf{f}_{(i)} - \mathbf{f}_{(i-1)}\right] \cdot \mu(\mathcal{H}_i) \end{aligned}$$

Because we can only find exactly k subsets $\mathcal{H}_i \subseteq \mathcal{K}$. We complete the proof.

□

In practices, the Choquet capacities are not given directly, but rather estimated empirically from the data. Therefore, to facilitate computations and interpret the Choquet capacities from different aspect, we shall define the following notions.

Definition 3.22 (Möbius and Zeta transforms [45])**.** Let $\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0}$. The Möbius transformation of μ, for $\mathcal{A} \subseteq \mathcal{K}$, is:

$$\mathcal{M}(\mathcal{A}) = \sum_{\mathcal{B}\subseteq\mathcal{A}} (-1)^{|\mathcal{A}\setminus\mathcal{B}|} \mu(\mathcal{B}) \tag{21}$$

The Zeta transform, or the inverse Möbius transform is given by:

$$\mu(\mathcal{A}) = \sum_{\mathcal{B}\subseteq\mathcal{A}} \mathcal{M}(\mathcal{B}) \tag{22}$$

Note that, $\mu(\mathcal{A}) \geq 0 \Rightarrow \sum_{\mathcal{B}\subseteq\mathcal{A}} \mathcal{M}(\mathcal{B}) \geq 0$, $\mathcal{M}(\emptyset) = 0$ and $\mu(\mathcal{K}) = 1 \Rightarrow \sum_{\mathcal{B}\subseteq\mathcal{K}} \mathcal{M}(\mathcal{B}) = 1$. There are a broad range of alternative representations of μ, see [45].

To further reduce the computational costs, one can relax how much the interactions among the members are modelled, and thereby, obtaining a sparse representation of capacities. One of the methods is to relaxing the non–additivity property. Generally, the capacities must be non–additive, so that the interactions across all subsets of members are taken into account. However, for computability, we shall consider up to $|\mathcal{A}| \leq \kappa, \mathcal{A} \subseteq \mathcal{K}$, where $\mu(\mathcal{A})$ is non–additive, and refer to it as κ–additive capacity.

Definition 3.23 (κ–additive capacity [45])**.** A capacity $\mu : 2^{\mathcal{K}} \mapsto \mathbb{R}_{\geq 0}$ is κ–additive, for $1 \leq \kappa \leq |\mathcal{K}|$ if for $\mathcal{A} \subseteq \mathcal{K} : \mathcal{M}(\mathcal{A}) = 0$ for $|\mathcal{A}| > \kappa$ and $\exists \mathcal{B} \subseteq \mathcal{K}, |\mathcal{B}| = \kappa : \mathcal{M}(\mathcal{B}) \neq 0$.

In our method, we use $\kappa = 2$, though one can freely adjust this hyperparameter for their need.

4 Choquet Adaptive Scalarisation (CAS)

4.1 Motivation

Before going into technical details, we would like to state reasons motivating this work. Note that, these are our expectations at the beginning of this work, and thus, it may not truly reflect the actual outcomes presented later in **Sec. 5**.

First, recent algorithms in adaptive scalarisation methods rely on solving several scalarised optimisation subproblems, and use the corresponding locally optimal solution to learn weights [27, 8, 28, 29, 30], and undoubtedly exhaust the computing budget. We do differently, each time, we solve one single scalarised optimisation subproblem, and retain all iterates, found by the nonlinear SOP solver, as training data to learn new weight. Indeed, each time, we also discard dominated solutions, and thus, attain a better Pareto front approximation. Doing this, we believe, after each iteration, we may discover different regions of the objective space, without the use of metaheuristics [11, 13], which are computationally expensive [22].

Second, linear scalarisation method cannot approximate non–convex PFs [4, 7, 9, 37], which are very common in real-world scenarios. Additionally, the presence of irregular and discontinuous PFs poses several challenges to scalarisation–based decomposition methods, including those built upon the *Chebyshev* function [12, 30]. Therefore, our application of Choquet integral aids in better approximation of non–convex and disconnected PFs, due to its nonlinearity [33], and may support the exploration capability in the adaptive scalarisation framework. Moreover, due to our implementation, the proposed algorithm can overcome the PS coverage difficulties at an expense of directly working on the decision space.

Third, ease of implementation. Adaptive scalarisation method is fairly easier than other classes of techniques, to implement, e.g. metaheuristics algorithms [11, 13] and PAL family [10, 15, 38].

4.2 Algorithm

Let us introduce a useful operator prior to our formalisation of the algorithm.

Definition 4.1 (Hadamard (element–wise) product)**.** Given two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$. The Hadamard product is defined as:

$$\mathbf{x} \odot \mathbf{y} = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} \odot \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \times \mathbf{y}_1 \\ \vdots \\ \mathbf{x}_n \times \mathbf{y}_n \end{bmatrix} \tag{23}$$

The core of our algorithm is the following scalarisation function.

Definition 4.2 (CAS scalarising/scalarisation function)**.** Given a Choquet capacity μ and a scaling vector $\mathbf{a}$, our scalarising function, $s_{\mu,\mathbf{a}} : \mathbb{R}^k \mapsto \mathbb{R}$, is given by:

$$s_{\mu,\mathbf{a}}(\mathbf{f}) \;=\; -k \cdot C_\mu \left(\mathbf{u}(\mathbf{f}) \odot \mathbf{a}\right), \qquad \mathbf{u}(\mathbf{f}) = \mathbf{z}^{\mathrm{ND}} - \mathbf{f}, \qquad \mathbf{f} \in \mathcal{F}, \tag{24}$$

where k is the number of objectives, $\mathbf{z}^{\mathrm{ND}}$ is the nadir of the current $\hat{\mathcal{F}}^*$, μ is empirically estimated from $\hat{\mathcal{F}}^*$, $\mathbf{a}$ is found by solving another NLP problem and C_μ is defined in **Def. 3.20**. The utility $\mathbf{u}$ turns the minimised objectives into quantities to be maximised, so that minimising $s_{\mu,\mathbf{a}}$ maximises the Choquet integral.

The objectives in our MOP may differ in magnitude/scales, and thus, $\mathbf{a}$ is a scaling vector, directly learnt from the current $\hat{\mathcal{F}}^*$. Likewise, the Choquet capacity $\mu(\cdot)$ is empirically estimated from the current $\hat{\mathcal{F}}^*$, by certain methods introduced in [45]. Let us formally propose the optimisation problem to find μ, $\mathbf{a}$ below.

Remarkably, in our method, we use the Möbius representation of the Choquet capacity μ, i.e. $\mathcal{M}(\mu)$, and hence directly compute $C_\mu(\cdot)$. Nonetheless, for brevity, we still use μ, and we also support, though not reported herein, estimation of μ and $\mathcal{M}(\mu)$, transformation between two representations and computation of Choquet integral using both. Within the scope of this report, we shall empirically estimate $\mathcal{M}(\mu)$ where μ is a 2–additive capacity and utilise it to scalarise multi–objectives via Choquet integral on $\mathcal{M}(\mu)$.

Definition 4.3 (In search of μ and $\mathbf{a}$ [45])**.**

$$\begin{aligned} \min_{\mathbf{a}\in\mathbb{R}^k} \quad & \sum_{\mathbf{f}\in\hat{\mathcal{F}}^*} \left[1 - k \cdot C_\mu\left(\mathbf{u}(\mathbf{f}) \odot \mathbf{a}\right)\right]^2 \\ \text{s.t.} \quad & \mu \leftarrow LP(\hat{\mathcal{F}}^*; p) \end{aligned} \tag{25}$$

where p is the penalty parameter, by default $p = \frac{1}{k}$ and $LP(\hat{\mathcal{F}}^*; p)$ is modelled externally as a linear program.

We use the `Nelder-Mead` in `SciPy` [46] to seek $\mathbf{a}$, and `linprog` (HiGHS) in `SciPy` to seek μ. Also, constraint $\mu(\mathcal{K}) = 1$ is removed in $LP(\hat{\mathcal{F}}^*; p)$ to reduce the computational cost. As this is beyond the scope of this project, we shall close the discussion here.

Definition 4.4 (CAS sub–SOP)**.** CAS solves the following sub–SOP at the t^{th} iteration:

$$\min_{\mathbf{x}\in\mathcal{X}} \; s_{\mu^{(t)},\mathbf{a}^{(t)}}\left(\mathbf{f}(\mathbf{x})\right) \tag{26}$$

where $\mu, \mathbf{a}$ are found by solving problem defined in **Def. 4.3**.

Every $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$ is retained, and later filtered to obtain $\hat{\mathcal{X}}^*$ and $\hat{\mathcal{F}}^*$ at the t^{th} iteration.

As reviewed earlier in **Sec. 2**, two design choices distinguish **Alg. 1** from the adaptive scalarisation methods. First, exactly one sub–SOP is solved per iteration, rather than a series of them. Second, the routine NLP returns every point $\mathcal{E}_t$ at which the nonlinear solver evaluates $\mathbf{f}$ before termination, which constitute the training data from which the next capacity is learnt. We offer the following diagram to demonstrate our algorithm workflow, i.e. **Alg. 1**.

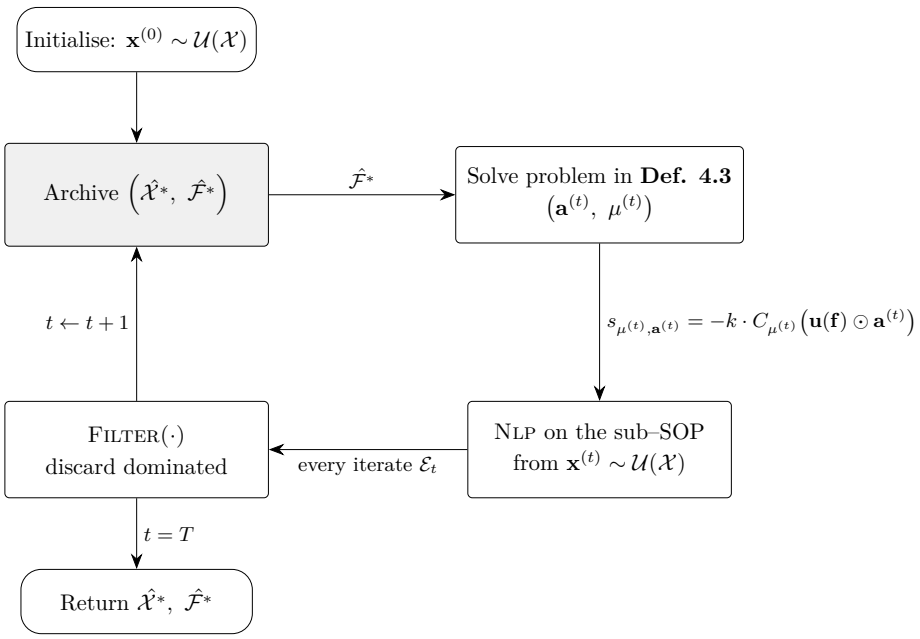


Figure 2: The flowchart of **Alg. 1**.

Algorithm 1 Choquet Adaptive Scalarisation (CAS)

Require: P_{MOP} over the box $\mathcal{X} = [\mathbf{x}^{\text{lb}}, \mathbf{x}^{\text{ub}}]$; budget T; tolerance τ; nonlinear solver $\textsc{Nlp}(\cdot)$
Ensure: Approximate Pareto set $\hat{\mathcal{X}}^*$ and Pareto front $\hat{\mathcal{F}}^*$

$\mathbf{x}^{(0)} \sim \mathcal{U}(\mathcal{X})$
$\mathcal{E}_0 \leftarrow \textsc{Nlp}\left(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}),\ \mathbf{x}^{(0)},\ \mathcal{X},\ \tau\right)$ ▷ μ not yet fitted: $\mathbf{w} = \mathbf{1}/k$
$\left(\hat{\mathcal{X}}^*, \hat{\mathcal{F}}^*\right) \leftarrow \textsc{Filter}\left(\mathcal{E}_0\right)$
for $t = 1, \ldots, T$ **do**
    Solve problem in **Def. 4.3** for $\left(\mathbf{a}^{(t)}, \mu^{(t)}\right)$
    $\mathbf{x}^{(t)} \sim \mathcal{U}(\mathcal{X})$ ▷ random fresh start
    $\mathcal{E}_t \leftarrow \textsc{Nlp}\left(s_{\mu^{(t)}, \mathbf{a}^{(t)}}\left(\mathbf{f}(\mathbf{x})\right),\ \mathbf{x}^{(t)},\ \mathcal{X},\ \tau,\ \texttt{max_iter} = 50\right)$ ▷ all retained
    $\left(\hat{\mathcal{X}}^*, \hat{\mathcal{F}}^*\right) \leftarrow \textsc{Filter}\left(\hat{\mathcal{F}}^* \cup \mathcal{E}_t\right)$ ▷ discard dominated points
end for
return $\hat{\mathcal{X}}^*,\ \hat{\mathcal{F}}^*$

function $\textsc{Filter}(\mathcal{Z},\ \textit{kind} = \text{“strongly”})$
    $\mathcal{P} \leftarrow \emptyset$ ▷ $\leq, <$ component–wise; $\epsilon = 10^{-5}$
    if *kind* = “strongly” **then**
        for $\mathbf{z} \in \mathcal{Z}$ in lexicographic order **do** ▷ Faster filtering for strongly Pareto optimal
            if $\nexists\ \mathbf{z}' \in \mathcal{P} : \mathbf{z}' \leq \mathbf{z}$ **then**
                $\mathcal{P} \leftarrow \mathcal{P} \cup \{\mathbf{z}\}$
            end if
        end for
    else if *kind* = “eps” **then**
        $\mathcal{P} \leftarrow \{\mathbf{z} \in \mathcal{Z} \mid \nexists\ \mathbf{z}' \in \textsc{Filter}\left(\mathcal{Z}, \text{“strongly”}\right) : \mathbf{z}' < \mathbf{z} - \boldsymbol{\epsilon}\}$ ▷ one copy per $\mathbf{z}$
    else if *kind* = “weakly” **then**
        $\mathcal{P} \leftarrow \{\mathbf{z} \in \mathcal{Z} \mid \nexists\ \mathbf{z}' \in \mathcal{Z} \setminus \{\mathbf{z}\} : \mathbf{z}' < \mathbf{z}\}$
    end if
    return $\mathcal{P}$ and $\mathbf{f}^{-1}\left(\mathcal{P}\right)$ ▷ $\mathbf{f}^{-1}(\mathcal{P})$ is already stored, not the inverse of $\mathbf{f}$
end function

function $\textsc{Nlp}(g,\ \mathbf{x}^{(0)},\ \mathcal{X},\ \tau,\ \texttt{max_iter})$ ▷ $g = s_{\mu,\mathbf{a}} \circ \mathbf{f}$ or $\mathbf{w}^\intercal \mathbf{f}$
    $\mathcal{E} \leftarrow \emptyset$
    repeat ▷ from $\mathbf{x}^{(0)}$
        one `L-BFGS-B` step on g from the current iterate, within $\mathcal{X}$
    until τ is met or `max_iter` is reached
    for all $\mathbf{x}$ at which the solver evaluates g **do**
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(\mathbf{x},\ \mathbf{f}(\mathbf{x}))\}$
    end for
    return $\mathcal{E}$
end function

In $\textsc{Filter}(\cdot)$, *lexicographic order* means sorting $\mathcal{Z}$ by the first objective, breaking ties by the second, and so on. We further note here that, $\textsc{Filter}(\cdot)$ filters the non–dominated solutions found up to the current iteration.

We note here, that $\mathbf{f}^{-1}$ implies the pre–image for $\mathcal{P}$, and in fact, recorded at every iterate in **Alg. 1**. In other words, it is not an inverse function of $\mathbf{f}$, but a manipulation of notation. For sake of benchmarking, we shall propose a baseline algorithm, by replacing the scalarisation

function of **Def. 4.2** by a linear combination $\mathbf{w}^\intercal \mathbf{f}(\mathbf{x})$, where $\mathbf{w}$ is updated by the current $\hat{\mathcal{F}}^*$.

Algorithm 2 Baseline algorithm

Require: P_{MOP} over the box $\mathcal{X} = [\mathbf{x}^{\text{lb}}, \mathbf{x}^{\text{ub}}]$; budget T; tolerance τ; nonlinear solver $\textsc{Nlp}(\cdot)$
Ensure: Approximate Pareto set $\hat{\mathcal{X}}^*$ and Pareto front $\hat{\mathcal{F}}^*$

$\mathbf{x}^{(0)} \sim \mathcal{U}(\mathcal{X})$
$\mathbf{w} \leftarrow \mathbf{1}/k \in \mathbb{R}^k$
$\mathcal{E}_0 \leftarrow \textsc{Nlp}\left(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}),\ \mathbf{x}^{(0)},\ \mathcal{X},\ \tau\right)$ ▷ uniform initial weights
$\left(\hat{\mathcal{X}}^*, \hat{\mathcal{F}}^*\right) \leftarrow \textsc{Filter}\left(\mathcal{E}_0\right)$
for $t = 1, \ldots, T$ **do**
    $\mathbf{w} \leftarrow \arg\min_{\mathbf{w} \geq \mathbf{0}} \left\| \hat{\mathcal{F}}^* \mathbf{w} - \mathbf{1} \right\|_2$ ▷ non–negative least squares
    $\mathbf{w} \leftarrow \mathbf{w} / \|\mathbf{w}\|_2$ if $\mathbf{w} \neq \mathbf{0}$, else $\mathbf{1}/k$ ▷ $\arg\min_{\mathbf{x}} \mathbf{w}^\intercal \mathbf{f}$ is scale–invariant
    $\mathbf{x}^{(t)} \sim \mathcal{U}(\mathcal{X})$ ▷ random fresh start
    $\mathcal{E}_t \leftarrow \textsc{Nlp}\left(\mathbf{w}^\intercal \mathbf{f}(\mathbf{x}),\ \mathbf{x}^{(t)},\ \mathcal{X},\ \tau,\ \texttt{max_iter} = 50\right)$ ▷ all retained
    $\left(\hat{\mathcal{X}}^*, \hat{\mathcal{F}}^*\right) \leftarrow \textsc{Filter}\left(\hat{\mathcal{F}}^* \cup \mathcal{E}_t\right)$ ▷ discard dominated points
end for
return $\hat{\mathcal{X}}^*,\ \hat{\mathcal{F}}^*$

For sake of understanding, we provide an informal conceptual illustration as follows.

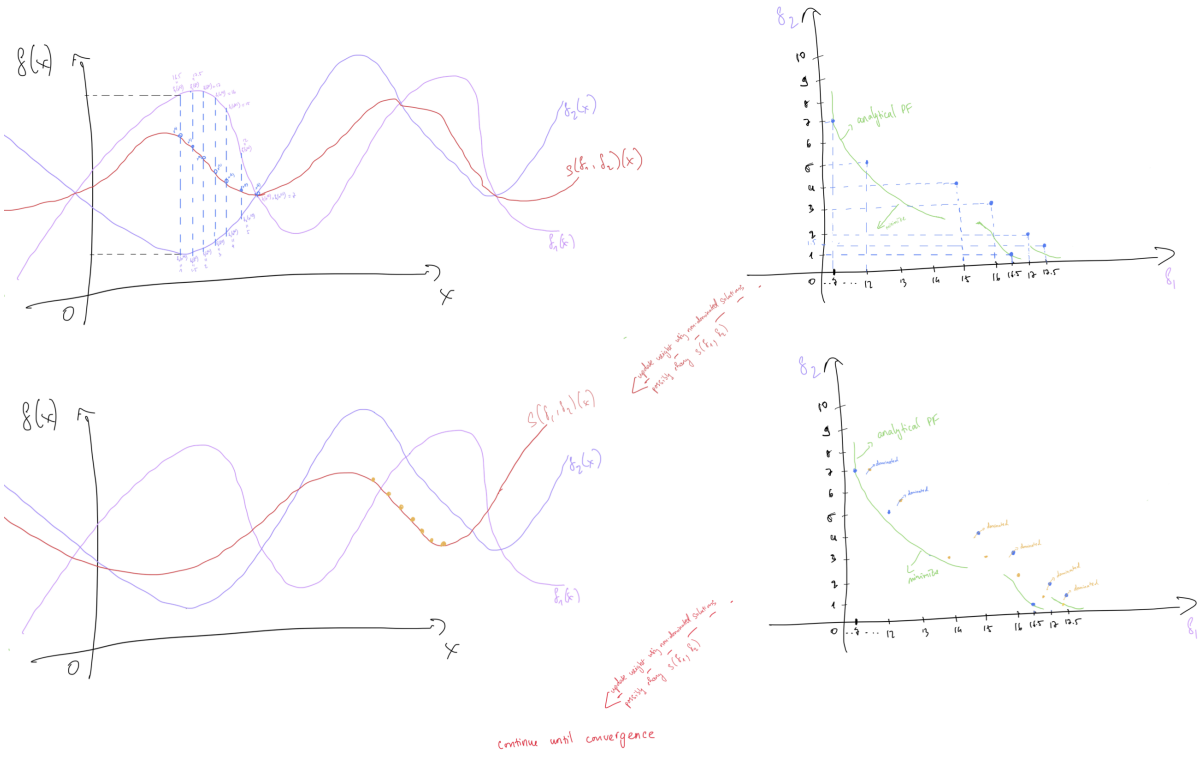


Figure 3: Sketch of the algorithm.

5 Experiments

5.1 Test problems

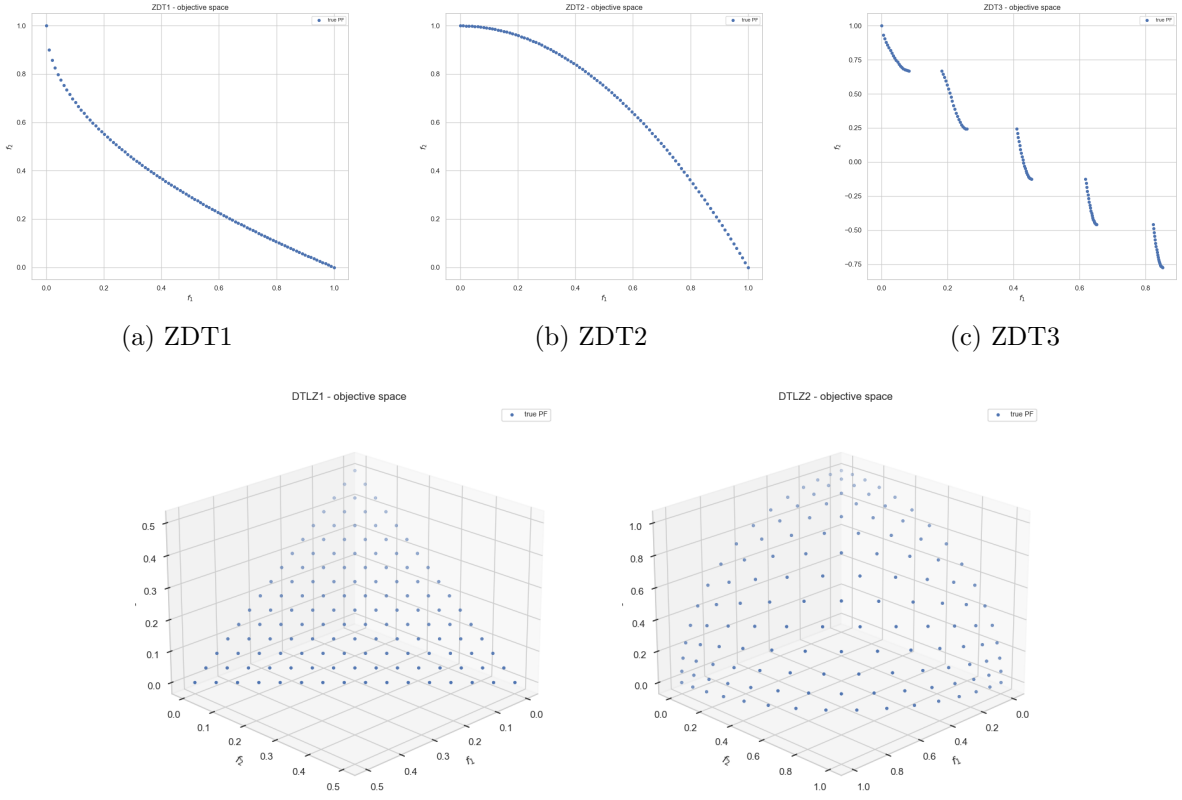
For convenience, we confine ourselves to box–constrained problems, rather than more complicated set of constraints. Nonetheless, it is viable to apply Penalty and Barrier methods or any available constrained NLP solvers, as given in [47, 48, 49], to handle the constrained cases. Generally speaking, we solve the following minimisation problem:

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{X}\subset\mathbb{R}^n} \quad & \mathbf{f}(\mathbf{x}) = \begin{bmatrix} \mathbf{f}_1(\mathbf{x}) & \cdots & \mathbf{f}_k(\mathbf{x}) \end{bmatrix} \\ \text{s.t.} \quad & \mathbf{x}^{\text{lb}} \leq \mathbf{x} \leq \mathbf{x}^{\text{ub}} \end{aligned}$$

We intentionally select test suites that the true PF can be found analytically, though this is generally untrue. We adopt the test problems implemented in `pymoo` [50], specifically three ZDT problems [51] and two DTLZ problems [52], all posed on the box $\mathcal{X} = [0, 1]^n$.

Problem	n	k	$\mathcal{F}^*$ shape
ZDT1 [51]	20	2	convex
ZDT2 [51]	20	2	concave
ZDT3 [51]	20	2	non–convex, disconnected
DTLZ1 [52]	10	3	convex
DTLZ2 [52]	10	3	non–convex

Table 1: Test suite problems.



(a) ZDT1 (b) ZDT2 (c) ZDT3

(d) DTLZ1 (e) DTLZ2

Figure 4: True PFs of the test problems of **Tab. 1**.

5.2 Performance metrics

In our experiment, we shall use four performance indicators, i.e. Hypervolume, number of function evaluations, number of non–dominated solutions found, and wall–clock time. More thoroughly, for number of function evaluations, we shall record the number of iterations in the NLP() until convergence, in **Algo. 1**, across all iterations T. Likewise, the number of non–dominated solutions is the size of the *approximate PF*, i.e. $|\hat{\mathcal{F}}^*|$. Since different scalarising strategy may produce either better or worse sub–SOPs, and thus, impact the termination of the solver used and the non–dominated solutions found. Although we leverage ϵ–Pareto optimality in **Alg. 1**, we re–filter the strongly Pareto optimal (as in **Def. 3.2**) solutions using all solutions found in the algorithm for comparison.

Regarding the wall–clock time (in seconds), we shall use the `time.perf_counter()` to estimate the time taken to exhaust the predefined budget T. Importantly, Hypervolume (HV) is used to quantify the explorability capability of the proposed algorithms, or the diversity of the non–dominated solutions. A more formal definition is presented as below.

Definition 5.1 (Hypervolume (HV) [53, 54])**.** Given $\hat{\mathcal{F}}^* \triangleq \{\mathbf{f}^{(1)}, \cdots, \mathbf{f}^{(m)}\} \subset \mathcal{A}_{\mathcal{F}}$ be the Pareto–optimal solutions found by an algorithm $\mathcal{A}$ and a reference nadir objective $\mathbf{z}_{\text{ref}}^{\text{ND}}$ defined by **Def. 3.9**. The Hypervolume (HV) is computed by:

$$HV(\hat{\mathcal{F}}^*, \mathbf{z}_{\text{ref}}^{\text{ND}}) = \mu\left(\bigcup_{i=1}^{m}\left[\mathbf{f}^{(i)}, \mathbf{z}_{\text{ref}}^{\text{ND}}\right]\right) \tag{27}$$

where $\mu(\cdot)$ is the Lebesgue measure and $\left[\mathbf{f}^{(i)}, \mathbf{z}_{\text{ref}}^{\text{ND}}\right] \triangleq \left\{\mathbf{z} \in \mathcal{F} \mid \mathbf{f}^{(i)} \preceq \mathbf{z} \preceq \mathbf{z}_{\text{ref}}^{\text{ND}}\right\}$ is the polytope enclosed by the non–dominated solution and the reference nadir point.

We leverage the HV computation function provided by `pymoo` [50].

5.3 Experimental setup

We benchmark CAS (**Alg. 1**) against the baseline (**Alg. 2**), which differs in the scalarising function and weight updating rules. Both are run on the test problems of **Tab. 1**, swept over n and k as set out in **Sec. 5.4**, under an identical configuration, as below.

Component	Setting	Remark
Budget T	50	outer iterations, i.e. number of sub–SOPs solved
Tolerance τ	10^{-6}	termination tolerance passed to NLP$(\cdot)$
FILTER$(\cdot)$	"strongly"	strongly Pareto optimal of **Def. 3.2**
NLP$(\cdot)$	`L-BFGS-B` [46]	box–constrained, gradient by finite differences
Capacity fit	`linprog` (HiGHS) [46]	inner problem of **Def. 4.3**
Outer search on $\mathbf{a}$	`Nelder-Mead` [46]	max. 100 iterations, $x_{\text{atol}} = 10^{-3}$
Baseline weights $\mathbf{w}$	`NNLS` [46]	non–negative least squares on $\hat{\mathcal{F}}^*$
Initial point	$\mathbf{x}^{(t)} \sim \mathcal{U}(\mathcal{X})$	resampled at every iteration, both algorithms
Random state	42	reproducibility

Table 2: Experimental configuration, held fixed across all problems and both algorithms.

Reference point. The HV of **Def. 5.1** requires a common $\mathbf{z}_{\text{ref}}^{\text{ND}}$ per problem. Hence, we select the worst non–dominated objective point of the union of both algorithms' fronts, so that the two algorithms are measured against the same polytope.

Reporting. Every metric is reported using a single run, and thus, results may deviate subject to computing resources and seed variance.

Environment. The experiments were conducted on:

- **Hardware**: MacBook M3 Pro - Total Number of Cores: 11 (5 Performance and 6 Efficiency) - Memory: 18 GB - System Firmware Version: 18000.120.36 - OS Loader Version: 18000.120.36
- **Software**: `Python` 3.11.14 using `NumPy` 2.3.5, `SciPy` 1.16.3, `pandas` 2.3.3, `pymoo` 0.6.2, `pyfmtools` 5.3.9, `Matplotlib` 3.10.8, `seaborn` 0.13.2, `Plotly` 6.9.0, and `Ipykernel` 7.1.0

Wall–clock time is therefore comparable within, but not across, machines.

5.4 Results

All results are produced by `pymoo_lab.ipynb`. The experiments sweep $n \in \{10, 20\}$ and $k \in \{2, 3, 4, 5\}$ for the DTLZ suite only, since the ZDT problems are bi–objective by default. Each problem scenario is solved under the same configuration specified in **Tab. 2**. We note here that, since $\mathbf{z}_{\text{ref}}^{\text{ND}}$ is re–computed for every problem, the performance metrics are comparable only row–wise.

Pareto front approximation. Let us visualise how close the algorithms (**Alg. 1** and **Alg. 2**) approximate $\mathcal{F}^*$ in the final run. We also offer an interactive plot in `pymoo_lab.ipynb` for those curious how each algorithm explores the objective space at each iteration. We shall structure the plots below as following. The first and second row represents Baseline (**Alg. 2**) and CAS (**Alg. 1**), respectively. The columns, from left to right, include ($\mathcal{F}^*$ - all visible evaluations - $\hat{\mathcal{F}}^*$), ($\mathcal{F}^*$ - $\hat{\mathcal{F}}^*$) and ($\mathcal{F}^*$ - $\hat{\mathcal{F}}^*$ - level set of the respective scalarising function).

The colours used may be confusing, we use gray and red to highlight $\mathcal{F}^*$ and $\hat{\mathcal{F}}^*$ in the first column, respectively. Meanwhile, we use red and blue to highlight $\mathcal{F}^*$ and $\hat{\mathcal{F}}^*$ in the second and third columns, respectively, owing to the presence of gray–coloured level curves.

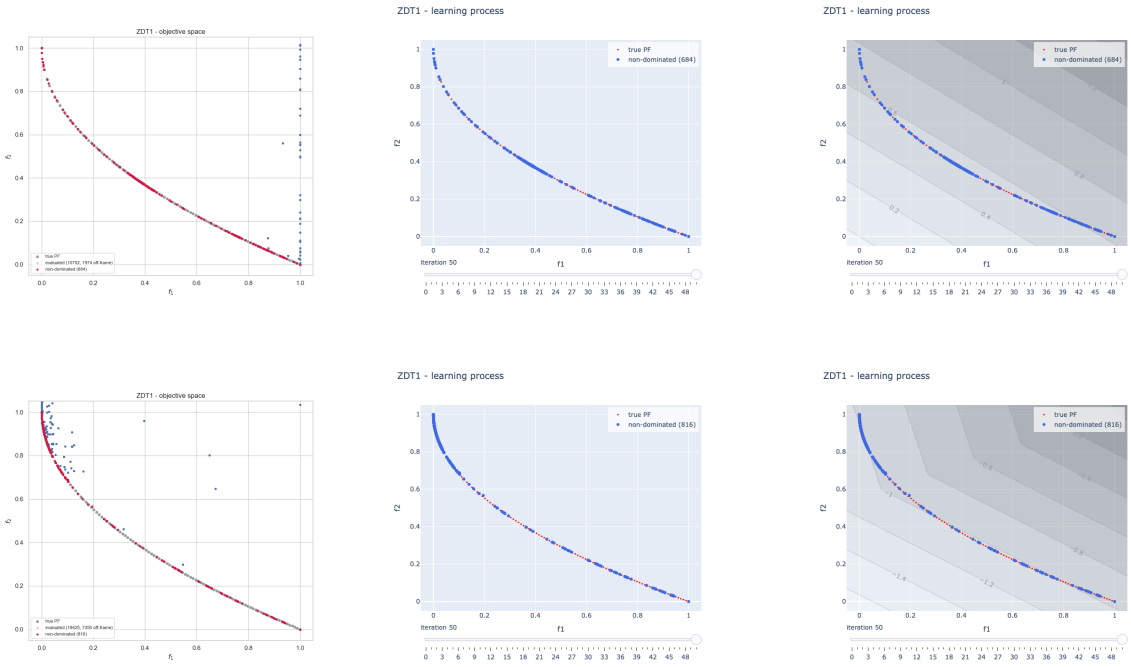


Figure 5: Visualisations of performance on ZDT1 ($n = 20, k = 2$, `max_iter` $= 50$).

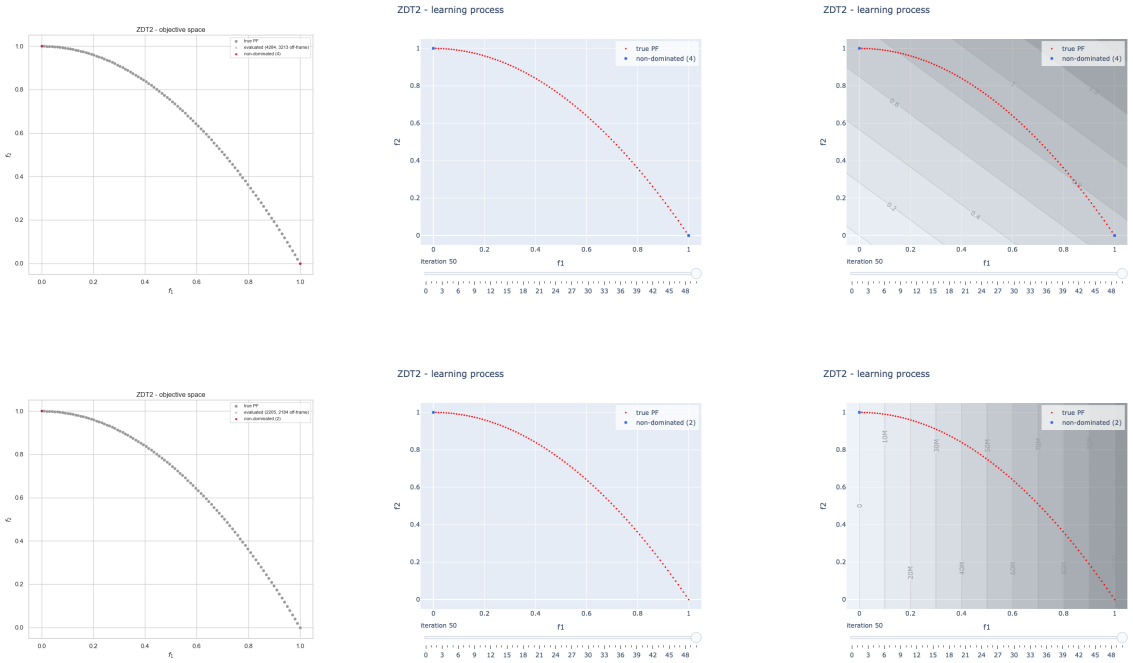


Figure 6: Visualisations of performance on ZDT2 ($n = 20, k = 2$, `max_iter` $= 50$).

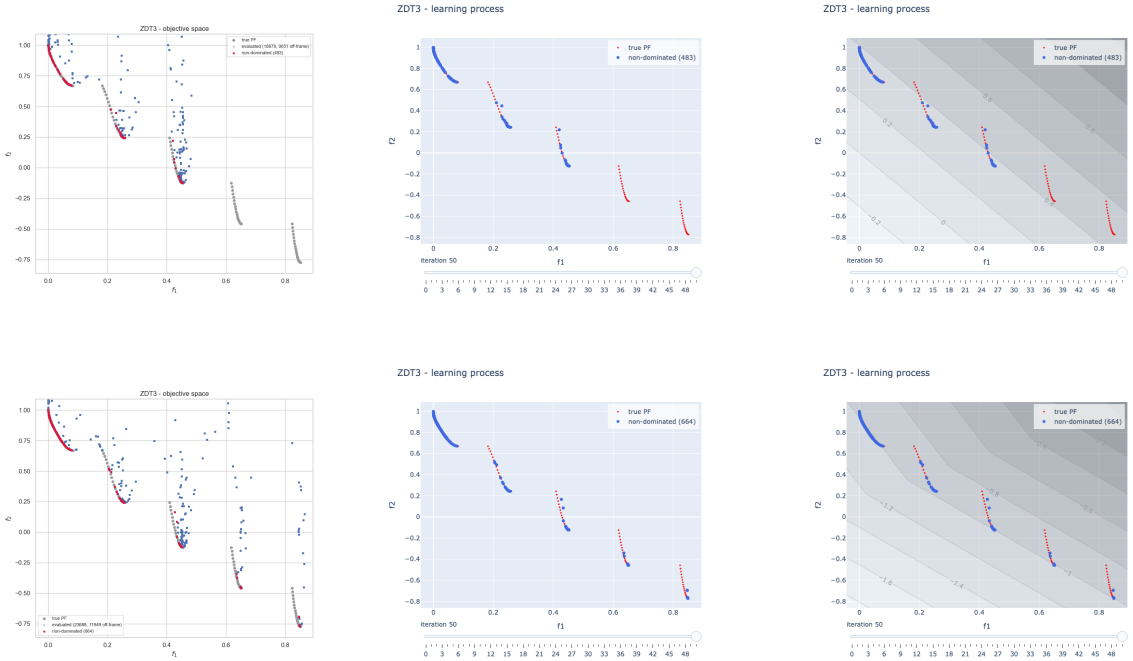


Figure 7: Visualisations of performance on ZDT3 ($n = 20, k = 2$, `max_iter` $= 50$).

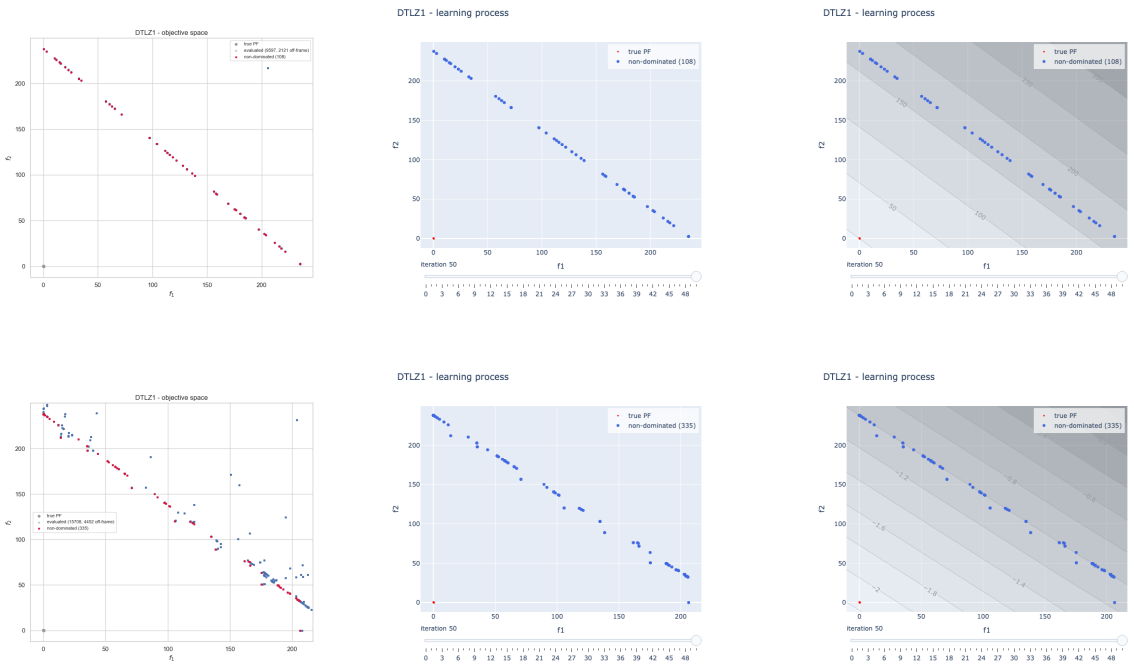


Figure 8: Visualisations of performance on DTLZ1 ($n = 20, k = 2$, `max_iter` $= 50$).

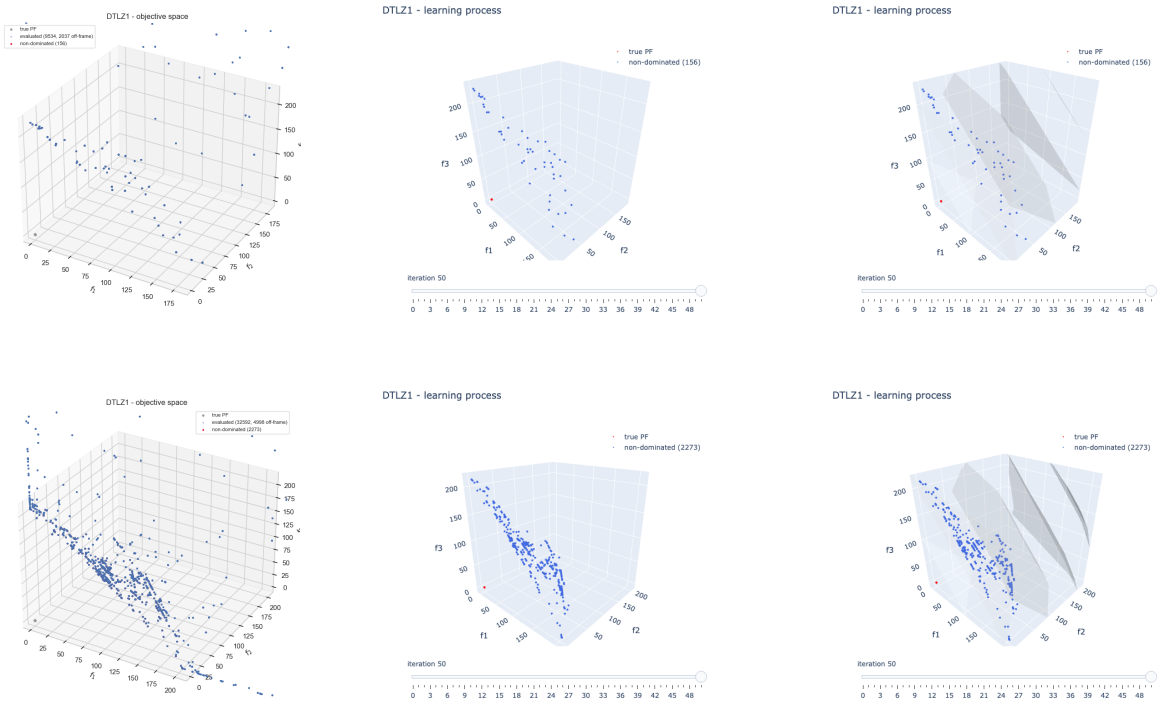


Figure 9: Visualisations of performance on DTLZ1 ($n = 20, k = 3$, `max_iter` $= 50$).

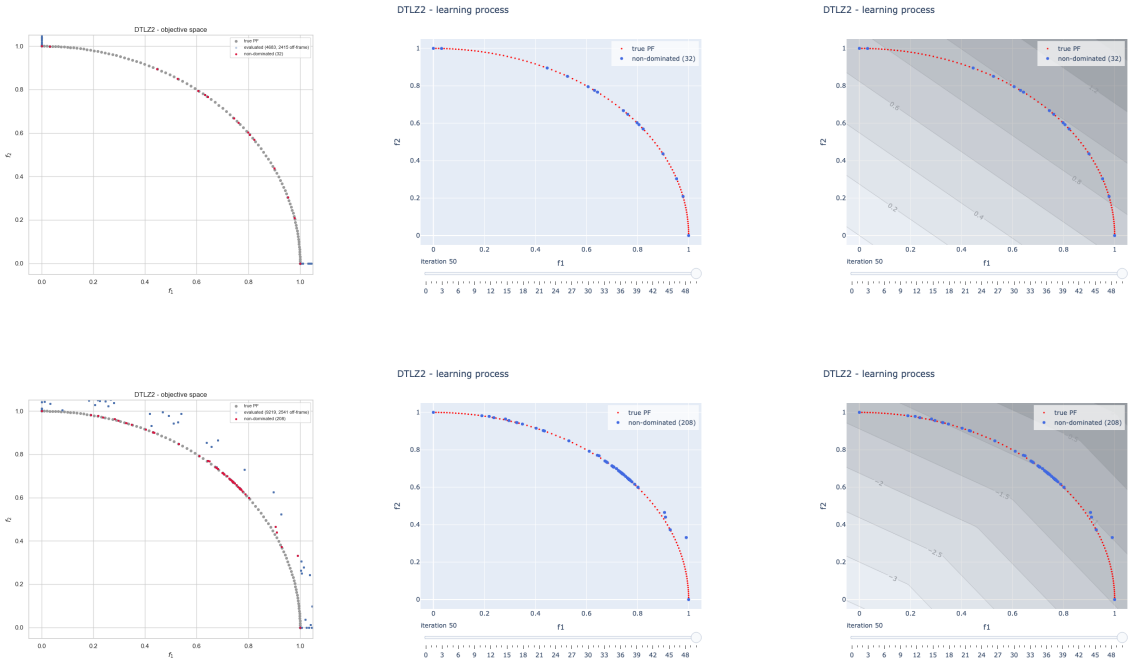


Figure 10: Visualisations of performance on DTLZ2 ($n = 20, k = 2$, `max_iter` $= 50$).

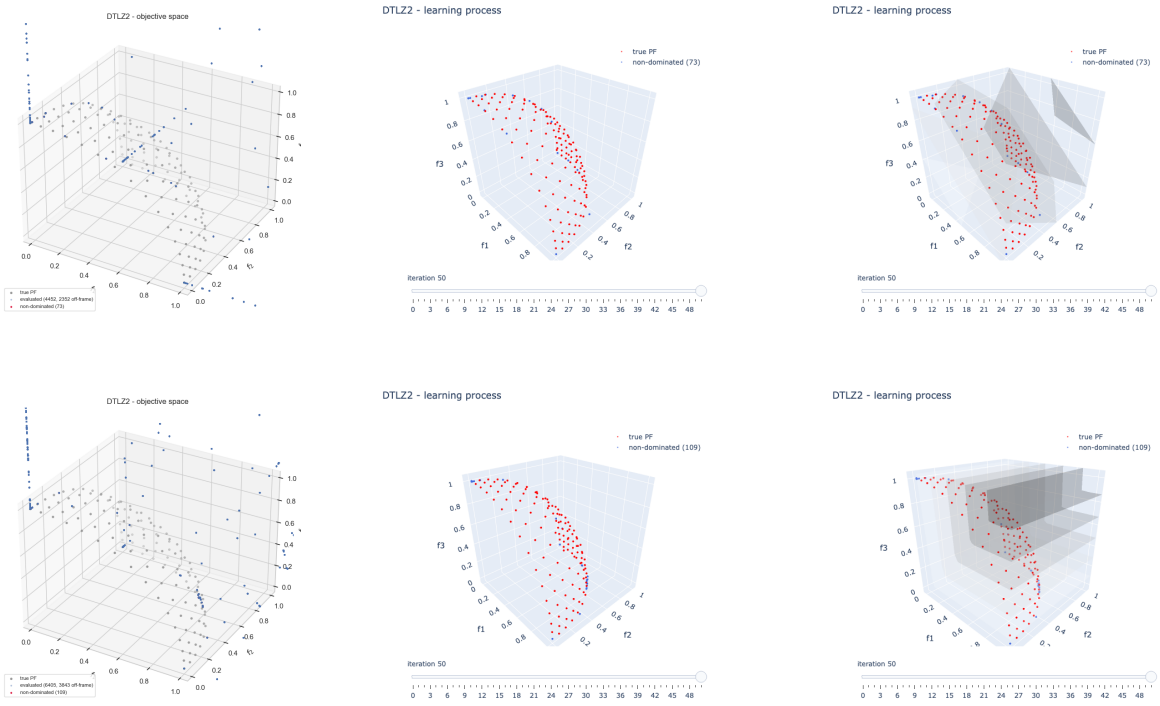


Figure 11: Visualisations of performance on DTLZ2 ($n = 20, k = 3$, `max_iter` $= 50$).

Approximate fronts. From **Fig. 5, 9, 8**, CAS covers the convex and linear Pareto fronts less evenly than Baseline, which reconstructs them almost fully. Noticeably, looking at the scale of the axes, both algorithms fail to reach close to the analytical $\mathcal{F}^*$ on DTLZ1. Interestingly, CAS outperforms baseline where the PF is non–convex and/or disconnected, evidently by covering

more of $\mathcal{F}^*$ in **Fig. 7, 10, 11**, but not on ZDT2 (**Fig. 6**). This supports our motivation earlier, i.e. non–linearity of Choquet integral attribute to generalising well to non–convex PF.

More thoroughly, on ZDT1 both algorithms trace the whole front. On ZDT2, whose front is non–convex, CAS keeps only 2 points near $(0, 1)$, whereas the baseline keeps 4, three near $(0, 1)$ and the last one at the end point $(1, 0)$. This agrees with the known weakness of the weighted sum, whose minimisers on a non–convex front lie at its extremes [4, 9]. On the disconnected ZDT3, of the five segments, the baseline populates only the first three, while CAS also reaches the fourth and fifth.

On DTLZ1, as said earlier, neither algorithm converges within the budget, i.e. the objectives of the archived points sum to about 100–115, while the true front is the simplex $\sum_i \mathbf{f}_i = 0.5$. The HV gap on DTLZ1 therefore reflects how widely the unconverged points spread, not how close they are to the PF. Interestingly, bi–objective DTLZ1 and tri–objective DTLZ1 influence how CAS approximate the corresponding PF, see **Fig. 8** and **Fig. 9**. On DTLZ2, the baseline produce fewer points lying on the surface of $\mathcal{F}^*$ than CAS, which may be due to non–linearity of scalarising function and its exploitation capability.

Table 3: Hypervolume and size of the approximate PF, the larger the better; gap is the relative difference. Bold marks the better of the two algorithms.

			HV			**Archive size** $\lvert\hat{\mathcal{F}}^*\rvert$	
Problem	n	k	**Baseline**	**CAS**	**gap (%)**	**Baseline**	**CAS**
ZDT1	10	2	**5.5010**	5.4947	0.115	778	**794**
ZDT1	20	2	**5.2592**	5.2540	0.098	684	**816**
ZDT2	10	2	6.2827	**6.4017**	1.860	30	**140**
ZDT2	20	2	5.5558	5.5558	0.000	**4**	2
ZDT3	10	2	**7.0562**	6.9512	1.488	**390**	37
ZDT3	20	2	5.4436	**5.6118**	2.998	483	**664**
DTLZ1	10	2	2.917×10^5	$\mathbf{2.954 \times 10^5}$	1.232	130	**171**
DTLZ1	10	3	9.421×10^7	$\mathbf{9.578 \times 10^7}$	1.638	180	**1037**
DTLZ1	10	4	8.438×10^9	$\mathbf{8.581 \times 10^9}$	1.664	288	**530**
DTLZ1	10	5	4.124×10^{11}	$\mathbf{4.200 \times 10^{11}}$	1.803	346	**427**
DTLZ1	20	2	1.440×10^6	$\mathbf{1.444 \times 10^6}$	0.300	108	**335**
DTLZ1	20	3	9.913×10^8	$\mathbf{9.984 \times 10^8}$	0.719	156	**2273**
DTLZ1	20	4	2.966×10^{11}	$\mathbf{2.991 \times 10^{11}}$	0.859	224	**2608**
DTLZ1	20	5	6.975×10^{13}	$\mathbf{6.981 \times 10^{13}}$	0.087	**837**	585
DTLZ2	10	2	9.718	**9.759**	0.415	16	**222**
DTLZ2	10	3	**11.572**	11.546	0.225	**73**	44
DTLZ2	10	4	39.394	**39.533**	0.352	233	**546**
DTLZ2	10	5	9.041	**9.184**	1.557	344	**2237**
DTLZ2	20	2	30.730	**30.739**	0.030	32	**208**
DTLZ2	20	3	**73.473**	73.434	0.054	73	**109**
DTLZ2	20	4	**693.408**	691.450	0.282	**179**	81
DTLZ2	20	5	**224.205**	223.407	0.356	**228**	190

Quality of the approximate PF. CAS attains the larger HV in 14/22 configurations (**Tab. 3**), though the marginal difference is insignificant, below 3%. CAS leads on all eight DTLZ1 configurations (0.09%–1.8%), on ZDT3 at $n = 20$ (2.998%), on ZDT2 at $n = 10$ (1.9%), and on 4/8 DTLZ2 configurations (0.03%–1.6%). The baseline leads on ZDT1 at both n (0.1% and 0.1%), on ZDT3 at $n = 10$ (1.5%) and on DTLZ2 with $n = 10$, $k = 3$ and $n = 20$, $k \in \{3, 4, 5\}$ (0.05%–0.4%), while ZDT2 at $n = 20$ is a tie.

Noticeably, on ZDT2 with $n = 20$ both archives $|\hat{\mathcal{F}}^*|$ shrink substantially. CAS keeps the larger $|\hat{\mathcal{F}}^*|$ in 16/22 configurations. Remarkably, larger $|\hat{\mathcal{F}}^*|$ does not imply larger HV; for instance,

on ZDT1 at $n = 20$, CAS keeps more points yet the baseline covers more volume.

Table 4: Function evaluations and wall–clock time. The lower the better.

			Function evaluations		**Wall–clock time (s)**	
Problem	n	k	**Baseline**	**CAS**	**Baseline**	**CAS**
ZDT1	10	2	5489	12309	1.85	51.63
ZDT1	20	2	10752	19425	3.61	52.35
ZDT2	10	2	1771	1881	0.67	9.04
ZDT2	20	2	4284	2205	1.68	27.72
ZDT3	10	2	8184	1914	3.21	9.72
ZDT3	20	2	18879	23688	15.08	84.69
DTLZ1	10	2	5016	5346	2.52	11.75
DTLZ1	10	3	4840	8602	1.33	119.04
DTLZ1	10	4	5115	5159	1.66	44.86
DTLZ1	10	5	5049	8096	1.84	178.50
DTLZ1	20	2	9597	15708	2.50	32.81
DTLZ1	20	3	9534	32592	2.86	546.31
DTLZ1	20	4	9555	33726	3.55	1219.92
DTLZ1	20	5	9849	10374	5.56	160.85
DTLZ2	10	2	2398	5368	0.55	6.17
DTLZ2	10	3	2563	2640	1.03	7.77
DTLZ2	10	4	2673	5401	0.89	69.27
DTLZ2	10	5	2926	9141	1.11	726.48
DTLZ2	20	2	4683	9219	0.85	9.88
DTLZ2	20	3	4452	6405	0.84	5.89
DTLZ2	20	4	4578	5103	1.10	11.57
DTLZ2	20	5	4347	5544	1.05	30.90

Cost. On function evaluations, CAS uses more evaluations in 20/22 configurations, evidently $\sim$0.2–3.5$\times$ that of the baseline. In one of the seven configurations won by the baseline (ZDT3 with $n = 10$), CAS spent fewer evaluations, signalling earlier termination of its sub–SOPs.

On wall–clock time, CAS is more computationally expensive, evidently $\sim$3.0–657$\times$ longer than the baseline. Most of this time goes into the problem of **Def. 4.3**: every Nelder–Mead step on **a** refits μ on the whole archive, so the cost grows with $|\hat{\mathcal{F}}^*|$ and with the capacity coefficients. Accordingly, the slowest run is DTLZ1 with $n = 20, k = 4$, which also keeps the largest $|\hat{\mathcal{F}}^*|$ (2608 points, 1219.92 s), whereas the fastest is on DTLZ2 with $n = 20, k = 3$ (109 points, 5.89 s).

6 Conclusion and discussion

Conclusion. In this paper, we introduced a novel algorithm, namely Choquet Adaptive Scalarisation (CAS), aiming to cover solutions spanning across irregular Pareto front shape, specifically succeeding to cover disconnected and non–convex presented in certain benchmarking test problems with different settings. We also benchmarked CAS against a linear version of the framework, and observed improvements in terms of Pareto front shape approximation and Hypervolume, at an expense of wall–clock time and number of function evaluations. In addition, we acknowledge the lack of theoretical analysis on convergence. Therefore, in future work, we shall seek an in–depth convergence analysis, and more importantly, extend the current algorithm to reduce computational costs while maintaining, or ideally enhance, its performance.

Limitations. The most critical and apparent of CAS is the computational overheads, evidently in **Tab. 4** and **Tab. 3**, CAS win at tiny marginal HV gaps ($< 3\%$), while it pays more on function evaluations ($\sim$0.2–3.5$\times$) and wall–clock time ($\sim$3.0–657$\times$).

Future work. We shall outline potential aspects of improvement as follows.

On the convergence analysis, as side experiments not presented herein, we plotted the change of models' weights after each iteration, but did not recognise any interesting patterns. Nonetheless, in the future work, we shall refer to other performance metrics, e.g. Hypervolume, to study the behaviours our CAS, which we hypothesise they may offer more insights.

Another fascinating aspect to discover is model's interpretability. Stemming from the interpretability of Choquet capacity [45], we hypothesise this inherent capability can be utilised to study the interactions among the objectives. Consequently, this may open more viewpoints to consider for algorithm extensions and MOP understanding.

In the algorithm, we implemented `Nelder-Mead` [46], which does not impose $\mathbf{a} > \mathbf{0}$, while Choquet integral requires non–negative capacity [45], and thereby it may influence the performance of our CAS. In the upcoming work, we shall inspect the impact of this solver and may replace with a constrained nonlinear solver, e.g. `L-BFGS-B` and `SLSQP` [46].

Empirically, from ablation studies, which are excluded herein, we observe that, alternating between **Def. 3.2, 3.4** and **Def. 3.3** with different choices of ϵ results in different computational and approximation performance of the models. We note here that, at the final evaluation stage, we consider strongly Pareto–optimal solutions. However, during the runs, different choices may influence the final evaluation outcomes. Hence, ablation studies are essential to understand its impact on the algorithm.

Exploration and exploitation is inherently implemented in our CAS. Again, by side experiments which are not shown herein, we found that, by tuning different termination criteria, mostly `max_iter` and `tolerance` τ, CAS can either explore or exploit from what it has learnt empirically, which also affect its computational complexity. Hence, we conjecture there is nice balance between exploration–exploitation trade–offs by alternating termination criteria. This requires more exploratory work to discover either prove or disprove our hypothesis proposed herein.

References

[1] Y. Chen, W. H. Chan, E. L. M. Su, and Q. Diao, "Multi-objective optimization for smart cities: A systematic review of algorithms, challenges, and future directions," *PeerJ Computer Science*, vol. 11, p. e3042, 2025.

[2] J. S. Angelo, I. A. Guedes, H. J. C. Barbosa, and L. E. Dardenne, "Multi-and many-objective optimization: Present and future in de novo drug design," *Frontiers in Chemistry*, vol. 11, p. 1288626, 2023.

[3] R. Liu, G. Ma, F. Kong, Z. Ai, K. Xiong, W. Zhou, X. Wang, and X. Chang, "Pareto-guided active learning for accelerating surrogate-assisted multi-objective optimization of arch dam shape," *Engineering Structures*, vol. 326, p. 119541, 2025.

[4] K. Miettinen, *Nonlinear Multiobjective Optimization*, vol. 12 of *International Series in Operations Research & Management Science*. Springer, 1998.

[5] R. H. Stewart, T. S. Palmer, and B. DuPont, "A survey of multi-objective optimization methods and their applications for nuclear scientists and engineers," *Progress in Nuclear Energy*, vol. 138, p. 103830, 2021.

[6] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Archives of Computational Methods in Engineering*, vol. 29, pp. 5605 – 5633, 2022.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] G. Eichfelder, "An adaptive scalarization method in multiobjective optimization," *SIAM Journal on Optimization*, vol. 19, no. 4, pp. 1694–1718, 2009.

[9] I. Das and J. E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 631–657, 1998.

[10] M. Zuluaga, G. Sergent, A. Krause, and M. Püschel, "Active learning for multi-objective optimization," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28(1) of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 462–470, PMLR, 17–19 Jun 2013.

[11] C. A. Coello Coello, "A comprehensive survey of evolutionary-based multiobjective optimization techniques," *Knowledge and Information Systems*, vol. 1, pp. 269–308, 1999.

[12] X. Zhou, X. Wang, and X. Gu, "A decomposition-based multiobjective evolutionary algorithm with weight vector adaptation," *Swarm and Evolutionary Computation*, vol. 61, p. 100825, 2021.

[13] M. A. Awadallah, S. N. Makhadmeh, M. A. Al-Betar, *et al.*, "Multi-objective ant colony optimization: Review," *Archives of Computational Methods in Engineering*, vol. 32, pp. 995–1037, 2025.

[14] S. Mahmood, H. Sun, A. Ali Alhussan, *et al.*, "Active learning-based machine learning approach for enhancing environmental sustainability in green building energy consumption," *Scientific Reports*, vol. 14, p. 19894, 2024.

[15] M. Zuluaga, A. Krause, and M. PÃ¼schel, "e-pal: An active learning approach to the multi-objective optimization problem," *Journal of Machine Learning Research*, vol. 17, no. 104, pp. 1–32, 2016.

[16] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *CoRR*, vol. abs/1012.2599, 2010.

[17] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison*, vol. 52, 07 2010.

[18] J. Li, Z. Chen, Z. Wang, and Y. chin Ivan Chang, "Active learning in multiple-class classification problems via individualized binary models," *Computational Statistics & Data Analysis*, vol. 145, p. 106911, 2020.

[19] J. Huang, R. Child, V. Rao, H. Liu, S. Satheesh, and A. Coates, "Active learning for speech recognition: the power of gradients," *CoRR*, vol. abs/1612.03226, 2016.

[20] M. Wu, C. Li, and Z. Yao, "Deep active learning for computer vision tasks: Methodologies, applications, and challenges," *Applied Sciences*, vol. 12, no. 16, 2022.

[21] F. Di Fiore, M. Nardelli, and L. Mainini, "Active learning and bayesian optimization: A unified perspective to learn with a goal," *Archives of Computational Methods in Engineering*, vol. 31, p. 2985–3013, Apr. 2024.

[22] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[23] G. Mavrotas, "Effective implementation of the ε-constraint method in multi-objective mathematical programming problems," *Applied Mathematics and Computation*, vol. 213, no. 2, pp. 455–465, 2009.

[24] D. E. Farrar and R. R. Glauber, "Multicollinearity in regression analysis: the problem revisited," *The review of economics and statistics*, vol. 49, no. 1, pp. 92–107, 1967.

[25] L. He, H. Ishibuchi, A. Trivedi, H. Wang, Y. Nan, and D. Srinivasan, "A survey of normalization methods in multiobjective evolutionary algorithms," *Trans. Evol. Comp*, vol. 25, p. 1028–1048, Dec. 2021.

[26] H. Ishibuchi, K. Doi, and Y. Nojima, "On the effect of normalization in moea/d for multi-objective and many-objective optimization," *Complexity Intelligence Systems*, vol. 3, pp. 279–294, 2017.

[27] I. Y. Kim and O. L. de Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and Multidisciplinary Optimization*, vol. 29, no. 2, pp. 149–158, 2005.

[28] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[29] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "MOEA/D with adaptive weight adjustment," *Evolutionary Computation*, vol. 22, no. 2, pp. 231–264, 2014.

[30] Z. Dong, X. Wang, and L. Tang, "Moea/d with a self-adaptive weight vector adjustment strategy based on chain segmentation," *Information Sciences*, vol. 521, pp. 209–230, 2020.

[31] Q. Gautier, A. Althoff, C. L. Crutchfield, and R. Kastner, "Sherlock: A multi-objective design space exploration framework," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, Mar. 2022.

[32] A. Deshwal, N. K. Jayakodi, B. K. Joardar, J. R. Doppa, and P. P. Pande, "Moos: A multi-objective design space exploration and optimization framework for noc enabled manycore systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, Oct. 2019.

[33] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*. Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2007.

[34] J. Branke, S. Corrente, S. Greco, R. Słowiński, and P. Zielniewicz, "Using Choquet integral as preference model in interactive evolutionary multiobjective optimization," *European Journal of Operational Research*, vol. 250, no. 3, pp. 884–901, 2016.

[35] M. Grabisch and C. Labreuche, "A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid," *Annals of Operations Research*, vol. 175, no. 1, pp. 247–286, 2010.

[36] M. Grabisch, "The application of fuzzy integrals in multicriteria decision making," *European Journal of Operational Research*, vol. 89, no. 3, pp. 445–456, 1996.

[37] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 3–18, 2018.

[38] P. Campigotto, A. Passerini, and R. Battiti, "Active learning of pareto fronts," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 506–519, 2014.

[39] G. Beliakov, L. Gao, Y. Xiang, and W. Zhou, "Multiobjective optimization by using cutting angle methods and hypervolume criterion," *Optimization Methods and Software*, vol. 40, no. 2, pp. 388–405, 2025.

[40] K. Deb, K. Miettinen, and S. Chaudhuri, "Toward an estimation of nadir objective vector using a hybrid of evolutionary and local search approaches," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 821–841, 2010.

[41] J. Guo, S. Yang, and S. Jiang, "An adaptive penalty-based boundary intersection approach for multiobjective evolutionary algorithm based on decomposition," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2145–2152, 2016.

[42] D. Denneberg, *Non-Additive Measure and Integral*, vol. 27 of *Theory and Decision Library B: Mathematical and Statistical Methods*. Dordrecht: Kluwer Academic Publishers, 1994.

[43] E. Pap, *Null-Additive Set Functions*, vol. 337 of *Mathematics and Its Applications*. Dordrecht: Kluwer Academic Publishers, 1995.

[44] Z. Wang and G. J. Klir, *Generalized Measure Theory*, vol. 25 of *IFSR International Series in Systems Science and Systems Engineering*. New York: Springer, 2009.

[45] G. Beliakov, S. James, and J.-Z. Wu, *Choquet Capacities and Fuzzy Integrals*. Theory and Applications of Computability, Springer Nature Switzerland, 2026.

[46] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[47] L. Grippo and M. Sciandrone, *Introduction to Methods for Nonlinear Optimization*, vol. 152 of *UNITEXT*. Cham: Springer International Publishing, 2023.

[48] O. Stein, *Basic Concepts of Nonlinear Optimization*. Berlin: Springer, 2024.

[49] N. Andrei, *Modern Numerical Nonlinear Optimization*, vol. 195 of *Springer Optimization and Its Applications*. Cham: Springer, 2022.

[50] J. Blank and K. Deb, "pymoo: Multi-objective optimization in python," *IEEE Access*, vol. 8, pp. 89497–89509, 2020.

[51] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[52] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary Multiobjective Optimization* (A. Abraham, L. Jain, and R. Goldberg, eds.), Advanced Information and Knowledge Processing, pp. 105–145, London: Springer, 2005.

[53] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *Trans. Evol. Comp*, vol. 3, p. 257–271, Nov. 1999.

[54] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, "Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point," in *Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms*, FOGA '09, (New York, NY, USA), p. 87–102, Association for Computing Machinery, 2009.

[55] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple Criteria Decision Making Theory and Application* (G. Fandel and T. Gal, eds.), vol. 177 of *Lecture Notes in Economics and Mathematical Systems*, pp. 468–486, Berlin, Heidelberg: Springer, 1980.

[56] A. Messac, A. Ismail-Yahaya, and C. A. Mattson, "The normalized normal constraint method for generating the Pareto frontier," *Structural and Multidisciplinary Optimization*, vol. 25, no. 2, pp. 86–98, 2003.

[57] B. Paria, K. Kandasamy, and B. Póczos, "A flexible framework for multi-objective Bayesian optimization using random scalarizations," in *Proceedings of the 35th Uncertainty in Artificial Intelligence Conference*, vol. 115 of *Proceedings of Machine Learning Research*, pp. 766–776, PMLR, 2020.

[58] R. Zhang and D. Golovin, "Random hypervolume scalarizations for provable multi-objective black box optimization," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, pp. 11096–11105, PMLR, 2020.

[59] T. Chugh, "Scalarizing functions in Bayesian multiobjective optimization," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, IEEE, 2020.

[60] G. Beliakov and N. Henseler, "Pyfmtools," Aug. 2024. Python library for handling and fitting fuzzy measures.

Appendix

A Notations and Terminology

Acronyms

Acronyms	Meaning
AI	Artificial Intelligence
AL	Active Learning
BO	Bayesian Optimisation
CAS	Choquet Adaptive Scalarisation (our method, **Alg. 1**)
DTLZ	Deb–Thiele–Laumanns–Zitzler test suite
HV	Hypervolume
KKT	Karush–Kuhn–Tucker (conditions)
L-BFGS-B	Limited-memory Broyden–Fletcher–Goldfarb–Shanno, with Bounds
lb, ub	lowerbound, upperbound
MOO	Multi–Objective Optimisation
MOOS	Multi–Objective Optimistic Search
MOP	Multi–Objective (Optimisation) Problem
NBI	Normal Boundary Intersection
NLP	Non–Linear Programs
NNLS	Non–Negative Least Squares
PAL	Pareto Active Learning
PBI	Penalty–based Boundary Intersection
PF	Pareto Front
PS	Pareto Set
SOP	Single–Objective (Optimisation) Problem; sub–SOP: the scalarised SOP solved at each iteration
WS	Weighted Sum
ZDT	Zitzler–Deb–Thiele test suite

Notation

Symbol	Meaning
General	
$\mathbb{R}$, $\mathbb{R}_{\geq 0}$	real and non–negative real numbers, $\mathbb{R}_{\geq 0} \triangleq [0, +\infty)$
$\triangleq$	defined as
inf - sup	infimum - supremum
$\odot$	Hadamard (element–wise) product
$\mathcal{U}(\mathcal{X})$	uniform distribution over $\mathcal{X}$

Symbol	Meaning
$\mathbf{B}_\delta(\mathbf{x})$	open neighbourhood of radius δ around $\mathbf{x}$
Multi–objective optimisation	
P_{MOP}	the MOP of **Def. 3.1**
n, k	dimension of the decision space, number of objectives
$\mathcal{K}$	index set of the objectives, $\mathcal{K} \triangleq \{1, \ldots, k\}$
$\mathbf{f}$	vector of single–objective functions, $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^k$
$\mathcal{X}$	feasible decision region/space
$\mathbf{x} \in \mathcal{X}$	(feasible) decision vector
$\mathbf{x}^{\text{lb}}$, $\mathbf{x}^{\text{ub}}$	lower and upper bounds of the box $\mathcal{X}$
$\mathcal{F}$	feasible objective region/space
$\mathbf{z} \in \mathcal{F}$	(feasible) objective vector
$\mathcal{X}^*$, $\hat{\mathcal{X}}^*$	true, approximate Pareto set
$\mathcal{F}^*$, $\hat{\mathcal{F}}^*$	true, approximate Pareto front
$\prec$	strong Pareto dominance, **Def. 3.2**
$\preceq$	weak Pareto dominance, **Def. 3.4**
$\preceq_{\boldsymbol{\epsilon}}$	$\boldsymbol{\epsilon}$–Pareto dominance with tolerance vector $\boldsymbol{\epsilon} \in \mathbb{R}^k$, **Def. 3.3**
$\mathbf{z}^{\text{ID}}$	ideal objective vector
$\mathbf{z}^{\text{ND}}$	nadir objective vector
$\mathbf{z}^{\text{ID}}_{\text{ref}}$, $\mathbf{z}^{\text{ND}}_{\text{ref}}$	reference ideal, nadir objective vector, estimated from the solutions found
$\mathcal{A}^{(i)}$, $\mathcal{A}^{(i)}_{\mathcal{F}}$	the i^{th} MOO algorithm, the objective vectors it has found
$\mathcal{A}^{\cup}_{\mathcal{F}}$	the objective vectors found by all algorithms considered, **Def. 3.9**
$s(\cdot)$	generic scalarising function, **Def. 3.10**
$s_{\mathbf{w},p}$, p	weighted L_p scalarising function and the order of its norm, **Def. 3.11**; $p = 1$ gives WS, $p = \infty$ Chebyshev
Choquet capacities and integral	
$2^{\mathcal{K}}$	power set of the objective indices
$\mathcal{A}$, $\mathcal{B}$	subsets of $\mathcal{K}$, i.e. elements of $2^{\mathcal{K}}$, at which μ is evaluated
μ	Choquet capacity (fuzzy measure)
C_μ	discrete Choquet integral w.r.t. μ, **Def. 3.20**
$\mathbf{f}_{\nearrow}$, $\mathbf{f}_{(i)}$	non–decreasing permutation of $\mathbf{f}$, its i^{th} component
$\mathcal{H}_i$	index set of the $k - i + 1$ largest components of $\mathbf{f}$
$g_{\mathcal{A}}(\mathbf{f})$	the coefficient $\mu(\mathcal{A})$ carries in $C_\mu(\mathbf{f})$, **Remark 3.21**
$\mathcal{M}$	Möbius transform of μ, **Def. 3.22**

Symbol	Meaning
κ	order of additivity of μ; $\kappa = 2$ herein, **Def. 3.23**
Algorithms	
$s_{\mu,\mathbf{a}}(\cdot)$	CAS scalarising function, **Def. 4.2**
$\mathbf{a}$	per–objective scaling vector
$LP(\cdot)$, p	the linear program fitting μ and its penalty parameter, $p = 1/k$ by default, **Def. 4.3**
$\mathbf{u}(\mathbf{f})$	utility $\mathbf{z}^{\mathrm{ND}} - \mathbf{f}$ maximised by CAS, **Def. 4.2**
$\mathbf{w}$	weight vector of the baseline, **Alg. 2**
T, t	budget (number of outer iterations), the current one
τ	termination tolerance of NLP$(\cdot)$
NLP$(\cdot)$	nonlinear solver (`L-BFGS-B`), returning every point at which it evaluates $\mathbf{f}$
$\mathcal{E}_t$	the pairs $(\mathbf{x}, \mathbf{f}(\mathbf{x}))$ evaluated by NLP$(\cdot)$ at iteration t, including its finite–difference probes
FILTER$(\cdot)$	non–dominated filter of **Alg. 1**
ϵ	tolerance of the "eps" branch of FILTER$(\cdot)$, $\epsilon = 10^{-5}$
$\mathcal{Z}$, $\mathcal{P}$	the objective vectors given to FILTER$(\cdot)$, and those it keeps
$\mathbf{f}^{-1}(\mathcal{P})$	stored pre–image (decision vectors) of the objective vectors in $\mathcal{P}$
$HV(\cdot,\cdot)$	hypervolume, **Def. 5.1**
m	number of points in an approximate PF